

IES Fernando Aguilar Quignon

2º ADMINISTRACIÓN DE SISTEMAS INFORMÁTICOS EN RED

PROYECTO FIN DE GRADO ASIR

Título del trabajo fin de grado

**Integración de la API de Directory a los formularios de
Google para la realización de partes de incidencia**



ALUMNO: José María Riol Sánchez

TUTOR: Javier García Estévez

CONVOCATORIA JUNIO CURSO ACADÉMICO 2022/2023

Índice

1. Estudio del problema y análisis del sistema	1
1.1. Introducción.	1
1.2. Objetivos.	2
1.3. Funciones y requisitos.	3
1.4. Planteamiento y evaluación de diversas soluciones.	4
1.5. Justificación de la solución elegida.	5
1.6. Modelado de la solución.	6
1.6.1. Recursos humanos.	6
1.6.2. Recursos hardware.	7
1.6.3. Recursos software.	8
1.7. Planificación temporal.	8
2. Ejecución del proyecto.	9
2.1. Estructura y códigos	9
2.1.1. Primera parte de la ejecución.	9
formIncidencia.gs	10
listaAlumno.gs	12
listaCurso.gs	13
index.html	14
2.1.2. Segunda parte de la ejecución.	16
index.html	17
actualizaUsuario.gs	18
buscaUsuario.gs	19
guardaCurso.gs	20
cargaCurso.gs	21
actualizaProfesores.gs	22
triggerEmail.gs	22
buscaProfesor.gs	24
2.2. Configuraciones.	25
3. Fase de pruebas.	26
3.0.1. Primeras pruebas	26
3.0.2. Segundas pruebas	27
3.0.3. Terceras pruebas	28
4. Documentación del sistema.	29
4.1. Introducción a la aplicación.	29
4.2. Manual de instalación.	30
4.3. Manual de administración.	35
4.4. Manual de usuario.	36
5. Acercamiento al apartado teórico del proyecto.	38
5.1. JavaScript	38
5.2. DOM	39
5.3. Centros de datos de Google	40
5.4. Google Forms.	40
5.5. Google Sheets.	41
5.6. Google Sites.	41
5.7. API de SDK de Admin.	42
5.8. API de directorio.	42
5.9. Forms Service	43
5.10. Clase SpreadsheetApp	43
5.11. Google Workspace	43
5.12. Google Apps Script	44
6. Conclusiones finales.	44

Índice de figuras

1.	Creación de un formulario de forma gráfica.	4
2.	Vincular respuestas a una hoja de cálculo.	6
3.	Registrar la cuenta del usuario que rellena el formulario.	6
4.	Ejemplo de la numerosa cantidad de formularios que se crean.	16
5.	Proyecto que tiene un contenedor.	16
6.	Creación de un proyecto con un contenedor.	17
7.	Habilitando el servicio de Admin SDK API.	25
8.	Permisos de la hoja de cálculo	26
9.	Consultando la lista de triggers	26
10.	Desplegables dentro del formulario.	27
11.	Página web simple.	27
12.	Filtrado de alumnos por el curso seleccionado.	28
13.	Hoja de cálculo con los cursos del instituto.	28
14.	Generación de formularios descontrolada.	29
15.	Esquema general del sistema.	29
16.	Crear proyecto a partir de un formulario.	30
17.	Lista de proyectos.	30
18.	El formulario es contenedor del proyecto.	31
19.	Lista de archivos necesarios.	31
20.	Lista de archivos necesarios.	31
21.	Indicando ID del formulario.	31
22.	Suministrando el ID al código.	32
23.	Suministrando la URL a cargaCurso.gs	32
24.	Suministrando la URL a guardaCurso.gs	32
25.	Vincular el formulario a una hoja de cálculo	33
26.	Indica a la página a donde debe redirigir	33
27.	Implementar el proyecto.	34
28.	Implementar el proyecto.	34
29.	URL de la aplicación web.	35
30.	Insertar la aplicación web en el sitio web.	35
31.	Página mostrada al acceder al enlace.	36
32.	Formulario resultante.	37
33.	Jerarquía de DOM.	39



1. Estudio del problema y análisis del sistema

1.1. Introducción.

Como en todos los colegios, a la hora de tener que reportar el mal comportamiento de un alumno se recurre a lo que se conoce como "parte de incidencia". Éste es un documento físico (papel) en el que se recogen ciertos datos, algunos de ellos son:

- Nombre completo del alumno que realizó la conducta inadecuada.
- El curso de dicho alumno.
- La fecha y la hora en la que se produjo la falta de conducta.
- La asignatura.
- Los motivos por el que el alumno ha tenido una conducta indebida.
- Observaciones.

Como es natural, un profesor puede escribir el nombre de un alumno de forma distinta a como lo haría otro profesor. Cogiendo como ejemplo mi nombre, podríamos encontrar un profesor que escriba "José María", otro profesor podría escribirlo como "Jose Mari", otra opción de escritura posible sería "José M^a", y así...

Esto tiene un problema, y es que a la hora de reunir los partes de un mismo alumno, el hecho de que el nombre de ese alumno esté escrito de formas distintas exige un trabajo extra para organizar los partes de dicho alumno que aparece con nombres distintos. Esto ralentiza mucho la labor de la jefatura de estudio del colegio convirtiendo un trabajo que podría realizarse en menos de una hora en una labor de varias horas.

Independientemente del formato del parte, el problema reside en la libertad de escribir de forma manual el nombre del alumno. Por tanto, tampoco se soluciona nada si hacemos uso de los Formularios de Google y el nombre del alumno se sigue escribiendo a mano. Sin embargo, a diferencia del papel, en el mundo digital podemos elegir un valor dentro de una lista desplegable evitando así que un usuario escriba por sí mismo el nombre y por tanto no tendríamos los problemas antes mencionados.

Son estas las facilidades que nos da el mundo digital que no aprovecharlas sería contraproducente, sobre todo en la actualidad en la que nos movemos. La facilidad para gestionar la información, administrarla, compartirla, extraer datos de ella es algo muy importante para el ser humano que se ha pasado toda la vida viviendo del conocimiento y de la información desde los inicios.

Tenemos que tener en cuenta que ya no es solo la facilidad que nos proporciona tratar con información que se almacena en servidores y en nuestros ordenadores. Es la facilidad que nos proporciona tener un sistema de directorios que se apoya en diferentes tipos de servicios que ofrece Google, pues significa que todos estos puntos estarán interconectados y el traspaso de información de un lado a otro o la extracción del mismo para volcarlo en otro lugar es mucho más cómodo y simple.

También es importante adaptarse a la evolución que ha tenido la tecnología conforme han pasado los años. Cada vez son menos los trabajos que se hacen en papel y menos los ficheros llenos de hojas de papel que se guardan en secretaría y en otras zonas del colegio habilitadas para el almacenamiento de documentos.

Ya no es solamente el hecho de tener los documentos almacenados en nuestros ordenadores de forma local, es el hecho de tener los documentos subidos a servidores haciendo posible a día de hoy su acceso desde cualquier lugar y desde cualquier dispositivo. Este aspecto también es muy importante a la hora de elaborar partes de incidencia, pues se puede imaginar el hecho de tener que realizar un parte y se deba de buscar dichos papeles fuera del aula, provocando un gran parón en el transcurso de la clase, evitando fluidez y espontaneidad. En cambio si podemos realizar el parte de conducta sin movernos del aula en un momento que podamos realizarlo se traduce en una mayor comodidad.



1.2. Objetivos.

Los Formularios de Google permiten crear preguntas que pueden ser respondidas de diferentes maneras y la que nos interesa es seleccionar la respuesta de una lista desplegable, como es en nuestro caso, dos desplegables, uno que liste los **cursos** y otro que liste los **alumnos** del curso seleccionado.

De esta forma cualquier profesor podrá completar el formulario y el nombre del alumno seleccionado no se verá alterado aunque reciba partes de diferentes profesores pues el nombre del alumno no se deja a libre albedrío para que cualquier usuario escriba el nombre como guste, si no que es el nombre tal y como se encuentra guardado en el dominio del instituto y éste permanece invariable.

Haciendo uso de los desplegables del formulario de Google se irán almacenando las respuestas (que no son más que los diferentes partes de conducta) y se evitará el problema de los nombre de los alumnos. Estos partes serán más fáciles de organizar y ahorrarán tiempo y trabajo pues al tener el nombre del alumno invariable por el desplegable y poder volcar la respuesta del formulario en una hoja de cálculo donde se podrá hacer búsquedas será sencillo tratar con la información aunque existan cientos de respuestas.

El motivo por el que se recurre a los formularios de Google y no a un formulario cualquiera (pues cualquier formulario puede contener menús desplegables) es aprovechar la estructura de directorio de usuarios que ofrece Google. Al fin y al cabo todos los servicios que ofrece Google como son "Documentos", "Gmail", "Hojas de cálculo", "Formularios"... están conectados en lo que se conoce Google Workspace permitiendo por ejemplo:

- Gestionar y administrar los datos de alumnos y profesores con facilidad.
- La subida de documentos, entregables por parte de los alumnos y la subida de temario, exámenes por parte de los profesores y quedando registrado dichas acciones u otras de la misma índole con su cuenta.
- Que se pueda crear un servicio de correo interno del instituto el cual también se puede configurar para permitir la entrada y salida de correo externo a la organización.

Es aprovechar la conexión existente entre todos los servicios de Google que posee el instituto y tener nuestro propio dominio con el que poder crear y administrar nuestras Unidades Organizativas como se hace con LDAP y de esta forma gestionar la información de los integrantes del dominio.

La información de los usuarios que se extrae del dominio del instituto es gracias a Google Directory el cual permite manejar y tratar dicha información como mejor nos convenga. Dentro de toda la información que nos pone a nuestro alcance la API Directory podemos ver por ejemplo qué integrantes tiene el colegio, quienes son alumnos, quienes profesores, averiguar el curso del alumno, indicar el tutor de cierto curso, temas más administrativos como los permisos que se dan a los usuarios que conforman las diferentes Unidades Organizativas, indica quien es administrador y quien no, el último logeo que tuvo el usuario en el sistema, el correo principal del usuario y uno de recuperación, cuando se creó la cuenta y diferentes campos más que nos ofrece.

Por todos los motivos que hemos listado se puede observar que elegir los formularios de Google para llevar el control de partes de incidencia es la mejor solución para el instituto, por tal y como está montado el sistema. Los profesores que rellenen los partes serán registrados pues utilizan la cuenta del instituto, los alumnos que se listan en el desplegable son con total certeza alumnos del colegio pues salen del directorio de usuarios del dominio.

En resumen, optar por los formmlarios de Google en un instituto cuya organización se respalda con un dominio alojado en Google y teniendo sus servicios una conectividad y compatibilidad tan grande es el objetivo a seguir. Todas las facilidades que nos ofrece la estructura de Google sobre todo por el hecho de poder establecer campos en el propio formulario de Google formado por datos que se pueden extraer de la API de Directorio y que las cuentas de los usuarios puedan ser controlados con los propios servicios de Google lo hacen la solución ideal para el objetivo que tiene este proyecto.



1.3. Funciones y requisitos.

Funciones.

Para el caso que nos atañe, los servicios que ofrecerá nuestro sistema son simples.

Nuestro sistema ofrecerá la posibilidad de hacer uso de un Formulario de Google con dos campos desplegables (**curso** y **alumno**) y una serie de campos explicando el motivo de la mala conducta del alumno y registrar dichas respuestas. Estas respuestas podrán vincularse con una "Hoja de cálculo" de Google que nosotros elijamos y volcar ahí la información del cuestionario rellenado y enviado con el fin de buscar en el futuro la información que nosotros deseemos.

Además de generar el formulario que será usado para los fines mencionados, nuestro sistema permite recoger datos del dominio de Google Workspace tales como los cursos, los alumnos y los profesores. En el caso de los cursos, se busca listar los cursos que existan actualmente en el instituto con la idea de ofrecer un desplegable de cursos totalmente actualizado. En el caso de los alumnos, se busca crear el desplegable con el nombre completo de los alumnos filtrado por el curso seleccionado y también actualizado. Por último, en el caso de los profesores, se busca poder enviar un correo al tutor del curso del alumno seleccionado y un correo al profesor que rellenó el parte.

También nuestro sistema hace uso de una "Hoja de cálculo" de Google que contiene la lista de cursos actualizada. Su cometido es permitir una lectura más rápida a la hora de generar la lista de cursos.

Si prestamos atención a lo que se usa, podemos ver que aprovechamos la conexión que tienen los diferentes servicios de Google, que en nuestro caso sería Google Forms, Hojas de cálculo de Google, Google Sites,... pues pasamos información de un lado a otro y hacemos uso de las funcionalidades que ofrece cada uno en nuestro beneficio con el objetivo de satisfacer la necesidad básica de este proyecto.

Por tanto estudiando bien el funcionamiento de nuestro sistema tenemos:

- Un formulario que recoge los datos del profesor que lo rellena, del alumno que cometió la incidencia, los motivos de la incidencia y la fecha del suceso.
- La hoja de cálculo que servirá para almacenar la lista de cursos. Dicha lista se escribe y se lee, por tanto podemos ver que hay movimiento de información entre diferentes servicios.
- Una hoja de cálculo donde se volcará las respuestas de los formularios. El cometido es poder revisar información de partes ya realizados con anterioridad, filtrar, ordenar y cualquier otra acción que se pueda realizar con las hojas de cálculo.
- Una aplicación web que se comunica con el resto de funciones del proyecto con el objetivo de servir una interfaz sencilla al usuario donde elegir el curso y mostrar el pertinente formulario. Además esta página permitirá desde la intranet del instituto que los profesores accedan con facilidad al formulario de incidencias.
- Hacemos uso de la API Directory para acceder a los datos de todos los alumnos del dominio del instituto y extraer la información que nos sea de utilidad para el caso de uso de nuestro proyecto.



Requisitos.

Para poder hacer uso del sistema es indispensable cumplir los siguientes requisitos:

- Disponer de una cuenta de Google del dominio de IES Fernando Aguilar Quignon (usuario@iesfernandoaguilar.es).
- Poseer los permisos necesarios para poder cumplimentar el formulario de parte de conducta.
- Tener acceso a la página web que muestra el formulario.
- Si se quisiera acceder, tener acceso a las hojas de cálculo. (tanto la hoja con los cursos como la hoja que recoge las respuestas del formulario).

En el caso de no cumplir algunos de los requisitos, el usuario podría no hacer un uso correcto del sistema.

1.4. Planteamiento y evaluación de diversas soluciones.

El primer planteamiento.

El principal planteamiento fue usar el servicio de Formularios de Google. De primeras se pensó crear un formulario con la interfaz gráfica que te ofrece Google para elaborar un formulario. De esta forma cualquier persona sin un conocimiento extenso en temas ofimáticos o de informática podría elaborar un nuevo formulario con el que poder recoger los datos necesarios en el parte de conducta.

Esta idea era clara desde el principio. Si queremos aprovechar todo el potencial que tiene Google Workspace, el formulario se tenía que crear con Google Forms pues este formulario podría aprovechar las ventajas que tiene que los servicios estén conectados unos con otros.

Pensar en hacer un formulario aparte, a primera vista no parece ser la solución que nos permita satisfacer las necesidades de nuestro cliente. Por ello se echó un vistazo al potencial que tiene la interfaz gráfica de Google Forms para crear un cuestionario desde cero.

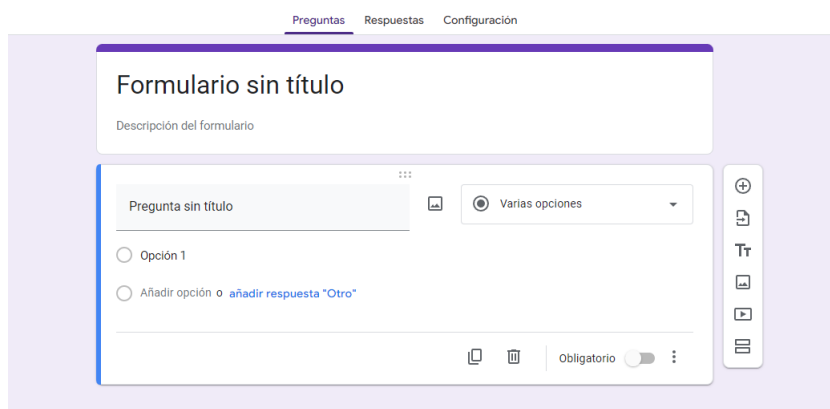


Figura 1: Creación de un formulario de forma gráfica.

Se podía crear todos los campos que eran necesarios para satisfacer las necesidades del formulario e incluso llevar a cabo ciertas configuraciones como recopilar direcciones de correo electrónico y enviar a los encuestados una copia de su respuesta. Sin embargo la lista desplegable por sí sola con lo que nos ofrece esta interfaz no podía mostrar la lista con los alumnos del colegio. Es por ello que crear un formulario de esta forma se nos quedaba corto, se precisaba más control sobre el mismo.

Hay cosas que una interfaz gráfica no es capaz de hacer, se necesitaba ese poder de personalización que solo se consigue programando. Es por ello que la idea del primer planteamiento se dejó aparcada y se empezó a pensar cómo sería posible crear un formulario más personalizado.



Segundo planteamiento.

Con la idea de necesitar un mayor control en los campos de nuestro formulario, después de un estudio de la documentación de Google se llegó a la conclusión de que un Formulario de Google se puede programar haciendo uso de *Google Apps Script*, ya sea para crearlo desde cero o para abrir un formulario ya existente y realizar cambios en él. Esto nos aproximaba más a lo que estábamos buscando pues para crear el desplegable con los alumnos se le podía pasar como valor un array de caracteres. Dicho array contendría el nombre de los alumnos.

De la misma forma se podía crear el desplegable de cursos mediante un array que contiene los cursos del instituto.

Esto ya parecía lo idóneo y estaban todos los problemas solucionados, pero existe un pequeño problema. En el formulario no había forma de hacer que el desplegable de alumnos cambiara según el curso seleccionado.

Actualmente los desplegables son independientes pues dentro del código son campos distintos sin ningún tipo de conexión. Buscando información se pudo barajar ciertas soluciones.

Había que buscar la manera de registrar un evento que al cambiar el curso la lista de alumnos cambie pero eso no era posible en el formulario de Google tal cual, pues los "triggers" usables en Google Forms eran *onOpen()* y *onFormSubmit()*, es decir al abrir el formulario o al enviarlo. *onOpen()* no nos sirve puesto que el desplegable de cursos lo cambiamos cuando el formulario ya se encuentra actualmente abierto y *onFormSubmit()* tampoco nos sirve porque ese disparador salta cuando ya confirmamos y enviamos el formulario. Podemos observar que ninguno de estos dos métodos para el "trigger" que queremos crear nos permite modificar el desplegable de cursos y que consecuentemente cambie el de alumnos.

Este problema impide tal y como está planteado el formulario que tenga el comportamiento que nuestro cliente desea. Por tanto había que buscar otra manera. En la búsqueda de solucionar este problema se dio con el tiempo con las dos posibles soluciones que se barajaron en este proyecto:

1. Crear el formulario completo en una página web.
2. Crear una aplicación web que pida el curso y abra un formulario ya existente.

Con la primera opción podríamos usar los *listener* que ofrece HTML con los que recoger el evento de un cambio en el desplegable de cursos y llamar a una función que actualice los valores del desplegable de alumnos. El resto de campos no serían complicados pues el contenido es estático y no hay que tratarlos de manera especial.

En cuanto a la segunda opción partiríamos de una aplicación web que nos listaría solamente los cursos del instituto. Una vez elegido el curso se nos abrirá un formulario de Google ya creado anteriormente con el desplegable de alumnos ya filtrado por el curso y el resto de campos que como pasa en la primera opción sus contenidos no dependen de otro campo, son estáticos. En este caso también habría un *listener* para que al cambiar el curso el desplegable de alumnos cambie.

1.5. Justificación de la solución elegida.

Tras barajar ambas opciones, se llegó a la conclusión que la mejor opción era la segunda.

El gran problema que tendríamos de hacer el formulario completo en una página web programada de cierta forma separada de la estructura de Google es que perderíamos funcionalidades propias de Google, ya que al aislar el formulario del propio servicio de Google Forms, estaríamos perdiendo buena parte de esa conectividad que tiene con el resto de servicios de Google Workspace. Eso hace que esta solución sea totalmente impensable, pues no debemos de prescindir de la comodidad y utilidad que Google nos proporciona. Algunas de esas funciones son el registro de autenticación de la cuenta de Google para rellenar el formulario y vincular las respuestas de los partes de conductas a una Hoja de cálculo de Google.



Claramente estas funciones se podrían programar de una forma u otra para simular su funcionamiento pero perderíamos algo de funcionalidad igualmente. No podemos esperar que nuestros fragmentos de código tengan las mismas funciones que los códigos de un equipo profesional de Google que trabaja específicamente para esto. Usando Google Forms nos aseguramos que todas las funcionalidades (tanto las mencionadas y obvias como las menos obvias) vayan sin ningún tipo de problema y se ejecuten de forma segura, es decir, no tengan vulnerabilidades. Además nos aseguramos la mayor compatibilidad posible al usar todo agrupado en un mismo sistema y no crear algo externo que se comunica con los servicios de la infraestructura de Google.

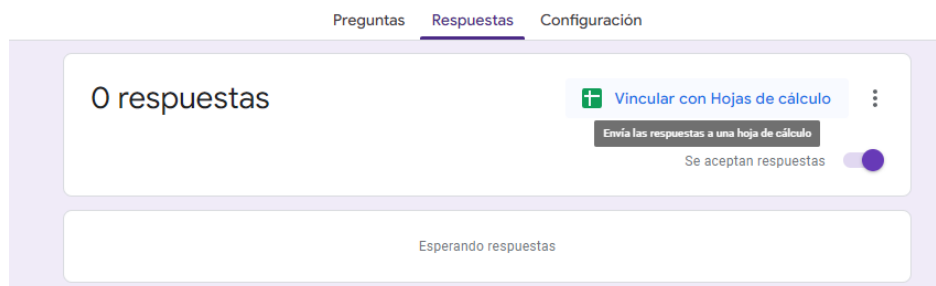


Figura 2: Vincular respuestas a una hoja de cálculo.

Figura 3: Registrar la cuenta del usuario que rellena el formulario.

Para poder disfrutar de todo lo que nos ofrece Google es recomendable hacer uso de Google Forms y mediante una página web, actualizar el listado de alumnos según el curso seleccionado, que se encontrará dentro del propio formulario de Google junto con el resto de campos. La página tal y como la queremos plantear siguiendo la solución elegida hace uso de los métodos de Google Form y Google Sheets y las funciones se llaman las unas a las otras para realizar cada una de las acciones esperadas en el proyecto para su correcto funcionamiento.

1.6. Modelado de la solución.

1.6.1. Recursos humanos.

Creación del proyecto

Hablaremos en primer lugar de las personas que hicieron posible la creación de este proyecto. Este proyecto nace con la idea de satisfacer una necesidad del cliente para una parte de su trabajo en el instituto. El cliente por tanto juega un papel importante en la creación del proyecto, pues sin su necesidad no haría falta plantear el sistema que estamos planteando para solucionar su problema.

En el otro lado me encuentro yo que soy la persona que desarrolla el sistema con el objetivo de solucionar el problema del cliente. Tras reuniones con el cliente y comentar el proyecto, propuesta



de soluciones y demás se fue elaborando poco a poco lo que a día de hoy es el sistema final. Tras estudiar las diferentes documentaciones con el fin de obtener respuestas a los obstáculos que encontraba por el camino el proyecto fue creciendo y acercándose al objetivo principal.

Muy importante también el papel del tutor del proyecto con el que me reunía semanalmente los viernes. Juntos hemos ido viendo la evolución del proyecto, compartía con él las posibles soluciones que tenía en mente, se veía si era un camino factible o había que rechazar esa idea, proporcionaba sus propias conclusiones para que yo en base a ellas actuara y siguiera con el desarrollo del proyecto, en definitiva el tutor del proyecto acompañó al proyecto en su desarrollo y fue testigo de su evolución.

Al igual que tenemos las personas que han hecho posible la creación del proyecto, debemos tener en cuenta las personas que hacen uso de él, de una forma u otra, pues sin ellas el proyecto no tendría sentido pues no sería útil.

Los profesores del colegio son los principales usuarios del sistema. Por desgracia tienen que rellenar partes de conducta en algunos momentos del curso porque un alumno ha tenido una conducta inadecuada. Tiene que ver también el hecho de que escriban el nombre manualmente de forma distinta, pues si dicho problema no pasara, este proyecto carecería de sentido igualmente.

También es importante el papel del administrador del sistema que hará mantenimiento del sistema a lo largo del tiempo para que siga existiendo. Sin un mantenimiento el sistema podría dejar de funcionar, dejaría de prestar servicios y por tanto dejaría de ser útil, es por ello que es necesaria su existencia también para que el proyecto siga con vida una vez ya esté implantado en el colegio.

1.6.2. Recursos hardware.

Podemos distinguir los dos lados de la arquitectura cliente-servidor.

Si partimos de la parte cliente los recursos hardware no son más que los diferentes dispositivos que utilizarán los distintos usuarios para poder acceder a la página web y poder hacer uso de la aplicación, de los servicios de Google... Estos dispositivos pueden ser móviles personales, tablets o los propios ordenadores del instituto. Se precisa que los dispositivos tengan conexión a Internet y dispongan de una cuenta de Google, de esta forma podrán hacer uso de nuestro sistema, porque recordemos que nuestro sistema está basado en el dominio del instituto, cualquier persona ajena al dominio le resultará imposible hacer uso de él. Con esto quiero decir que los dispositivos deben de cumplir ciertos requisitos para que sean candidatos a poder usar el sistema, pues no cualquiera debe de poder acceder al servicio que este sistema ofrece.

En cuanto a la parte servidor, es muy difícil decir con exactitud que servidores nos están ofreciendo servicios, pues el sistema que ofrece Google es un sistema distribuido. Podemos tener parte de nuestros datos en un servidor y otros datos en otro lugar del mundo. Estos servidores están interconectados unos con otros para ofrecer ciertas ventajas propias de estos sistemas. Una cosa está clara y es que Google de forma distribuida nos ofrece todos los servicios que nos ofrece y nos permite hacer uso de nuestro sistema de una forma fácil y rápida, además de poderlo hacer en cualquier parte y en cualquier momento, algo muy importante a día de hoy. Todos los datos se almacenan en discos que se encuentran en los Centros de datos de Google y sus servidores nos prestan los servicios que proporcionan a través de sus herramientas, de las cuales nosotros hacemos uso y aprovechamos para el funcionamiento de nuestro proyecto.



1.6.3. Recursos software.

Nuestro sistema hace uso de diferentes servicios y programas que nos facilita Google. Nosotros entendiendo como funcionan hacemos uso de ellos de la forma que mejor nos convenga. Estudiando cada uno de los servicios de los cuales hace uso nuestro sistema, los recursos software que usa nuestro proyecto son:

- Formularios de Google. Permite crear formularios online, enviar encuestas, se puede analizar los resultados de las respuestas, además de facilitar todo lo mencionado desde cualquier lugar.
- Hojas de cálculo de Google. Permite crear hojas de cálculo colaborativas en tiempo real y desde cualquier dispositivo. Permite además una conexión fluida con otras aplicaciones de Google.
- Google Sites. Permite crear y editar sitios web sin esfuerzo para nuestros equipos. También es colaborativo y permite acceder de forma sencilla a todo nuestro contenido.
- API Directory. Permite crear y administrar de forma programática recursos controlados por el administrador que son propiedad de una cuenta de Google Workspace.
- Google Workspace. Servicio de Google que proporciona varios productos que pueden ser personalizados por el cliente. Cuenta con varias aplicaciones web con funciones similares a las que puede ofrecer Microsoft Office o Libre Office.
- API de SDK de Admin. Es una colección de interfaces RESTful que permite a los administradores gestionar organizaciones de Google Workspace a gran escala.
- Lenguaje de programación JavaScript. El lenguaje utilizado para el desarrollo del proyecto en Google Apps Script es el de JavaScript, muy útil y usado para el entorno en el que se mueve nuestro proyecto.
- HTML. Para la pequeña interfaz Web tenemos un archivo.html que llama a las diferentes funciones JavaScript con el objetivo de servir al usuario el formulario que debe de usar para rellenar el parte de conducta.

1.7. Planificación temporal.

La planificación que se ha hecho de este proyecto fue la siguiente:

- Investigación para dar con las posibles soluciones al problema planteado. Esto es buscar la manera de evitar que los profesores a la hora de introducir un nombre se haga de forma manual y de lugar a variaciones del nombre de un mismo alumno. Para ello se pensó rápidamente en el uso de formularios de Google con el objetivo de servir el nombre de los alumnos mediante un desplegable. Sin embargo con las herramientas actuales y sin programación era insuficiente.
- Realización de un curso de JavaScript en Codecademy. Google Apps Script hace uso de JavaScript para poder escribir código. Al ser un lenguaje no visto en el ciclo fue necesario realizar un curso online para aprender las nociones básicas e intermedias de este nuevo lenguaje (sintaxis, lógica, estructura de datos, bucles, funciones, clases, objetos...). Este curso permitió comprender e interpretar el código que de mostraba de ejemplo en cada una de las documentaciones de los servicios de Google y poder aprovechar sus características.
- Una vez estudiado el curso de JS se procedió a realizar el siguiente paso, hacer el formulario pero programando. Para ello se estudió la documentación de "Service Form" con el objetivo de saber cómo se puede programar un formulario de Google. Esto solventó algunos de los problemas iniciales pero seguía sin ser suficiente pues seguía pendiente como hacer el desplegable de alumnos y de cursos, pues con lo que nos ofrecía Service Form no había manera de obtener dichos datos del dominio, había que llegar aún más lejos.
- Se estudió cómo sacar el desplegable de cursos. Tras un tiempo de estudio se llegó a la conclusión que había que hacer uso de la API de SDK de Admin. Ofrecía diferentes métodos con sus respectivos campos que solucionaban el cómo extraer la información de los alumnos. Conforme pasaba el tiempo se fue trasteando con la API para ver cómo acceder al campo de



Departamento y poder devolver diferentes campos del usuario como pueden ser el apellido y el nombre. No era suficiente con saber qué métodos usar y cómo programarlo, era necesario activar el servicio de la API desde el proyecto para poder extraer la información de nuestro dominio y el intérprete lo entendiese correctamente.

- Una vez estaba claro cómo extraer la información de los usuarios del dominio se tuvo que proceder a listar todos los usuarios del dominio. En un principio era imposible listar todos los usuarios, pues el máximo número de resultados que la API arrojaba eran quinientos usuarios. Esto hacía imposible recorrer la lista entera de usuarios del dominio pues el dominio se conforma actualmente por más de mil usuarios. Se estudió la posible solución y se llegó a la conclusión de que era necesario el uso de un pageToken. Paginando dichos resultados se logró llegar a listar todos los usuarios del dominio. Así es posible sacar el curso de cada alumno y haciendo ciertos arreglos obtendríamos la lista de todos los cursos que se imparten en el instituto.
- Pudiendo listar ya todos los usuarios sin problemas comenzamos con el planteamiento de diferentes soluciones para el problema de la dependencia del desplegable de alumnos. Tras plantear un par de soluciones se desarrollaron ambas y se vieron los pros y contras de cada una de ellas llevando a la elección del más conveniente. Tras ver los problemas de aislar el formulario del propio servicio de Google Forms se decidió optar por usar una aplicación web que sirva el formulario usando Google Forms. De esta forma podemos hacer que la lista de alumnos cambie según el curso elegido y sigamos disfrutando de las ventajas de Google Forms.
- Posteriormente a tener ya la aplicación web sirviendo el formulario, se procedió a la adaptación de la solución elegida para satisfacer las necesidades del cliente. Hay ciertos aspectos que el cliente quería sí o sí y se dedicó tiempo para que estos aspectos se cumplieran. Como por ejemplo evitar que se generasen nuevos formularios cada vez, pues la generación de un formulario por cada vez que se quisiera rellenar un parte volvería a convertir la labor del jefe de estudios en algo muy tedioso y difícil de controlar. No nos podíamos quedar aquí, teníamos que avanzar un poco más para buscar una solución a este problema.
- Tras darle una vuelta de hoja al proyecto y plantear la solución de otra manera, se logró evitar que se generasen formularios de forma masiva. Con eso ya aparcado quedaba solamente los retoques y mejoras diversas. Pequeños cambios que no alteran el principal funcionamiento del proyecto pero son necesarias para mejorar el estado del mismo o satisfacer pequeñas necesidades extras del cliente, como puede ser cambiar el como se muestra el nombre completo de los alumnos en el desplegable, el envío automático de correo a diferentes cuentas mediante triggers, mejoras en el estilo de la página web que sirve el desplegable y el formulario, etc...

2. Ejecución del proyecto.

2.1. Estructura y códigos

Se va a detallar a continuación cómo se estructura el proyecto a nivel técnico.

2.1.1. Primera parte de la ejecución.

Este proyecto se compone de una serie de archivos.gs y un archivo.html que contiene el código para según que función contenga realizar diferentes acciones como la creación del formulario, filtrar alumnos, listar cursos, etc... Tiene dos partes. La primera parte es la solución que se pensó al principio del mismo que es creando un formulario desde cero con código y contiene los siguientes archivos:

- formIncidencia.gs
- listaAlumno.gs
- listaCurso.gs
- index.html



formIncidencia.gs

En esta función podemos encontrar el código correspondiente a la creación de un formulario y sus diferentes campos (desplegables, títulos, opciones marcables...). Para ello se crean la variable que permite la creación del formulario

```
var form = FormApp . create ( ' Parte de Incidencia ' ) ;
```

Y las diferentes variables que crean cada uno de los campos, ya sean una lista, una serie de opciones de checkbox, un campo para introducir la fecha y hora del suceso...

```
var cursitos = form . addListItem ( ) ;  
var cursitos = form . addListItem ( ) ;  
...
```

Para poder generar la lista de alumnos según el curso elegido se llama a la función

```
listaAlumno(cursoelegido)
```

Esta función devuelve un array que se le pasa a la variable alumnos que es una variable de tipo form.addListItem(), es decir un objeto del formulario de tipo Lista. Luego con los métodos adecuados se crea el título de ese campo del formulario y la propia lista desplegable.

Posterior a esto se encuentran los diferentes campos estáticos del formulario como son la hora, los checkbox para elegir qué conductas contrarias tuvo el alumno y siguen la misma dinámica de crear el título y crear cada una de las opciones.

En las líneas 78 y 79 podemos encontrar las URLs que se generan para poder acceder al formulario tanto para la vista normal que permite rellenar el formulario como la vista de edición para modificar los campos del formulario.

Luego tenemos la creación de un trigger y la función

```
onFormSubmit()
```

Esta función rescata las respuestas dadas en el formulario en cada uno de los campos del mismo y las almacena en una variable llamada *emailBody*, variable que se pasa a la función

```
sendEmail()
```

que tiene como finalidad enviar el correo a las diferentes personas acordadas con el cliente para tener una copia de las respuestas del formulario.

```
1 //Funcion que crea el formulario de Google con los campos necesarios para rellenar  
  el parte de incidencia.  
2 function parteIncidencia(cursoelegido) {  
3  
4   //Declaracion de las variables inicializadas con las funciones del formulario  
5   //para crear el formulario y los distintos campos.  
6   var form = FormApp.create('Parte de Incidencia');  
7   var cursitos = form.addListItem();  
8   var alumnos = form.addListItem();  
9   var fechahora = form.addDateTimeItem();  
10  var titulo = form.addSectionHeaderItem();  
11  var item = form.addCheckboxItem();  
12  var item2 = form.addCheckboxItem();  
13  var item3 = form.addCheckboxItem();  
14  var item4 = form.addCheckboxItem();  
15  var item5 = form.addCheckboxItem();  
16  try {  
17  
18   //Llama a la funcion listaAlumno() pasando como parametro  
19   //el curso elegido en la pagina web y almacena los valores en el vector  
20   //alumL. Posteriormente se crea el titulo y el menu desplegable  
21   //con la lista de alumnos del instituto filtrados por el curso elegido  
22   const alumL = listaAlumno(cursoelegido);  
23   alumnos.setTitle('Eliza el alumno infractor');  
24   alumnos.setChoiceValues(alumL);  
25   console.log("ID de alumnos: "+alumnos.getId());
```



```
26
27 //Establece un campo para introducir la fecha y la hora de la incidencia.
28 fechahora.setTitle('Fecha y hora de la incidencia');
29
30 //Establece el titulo y las opciones marcables del campo item
31 titulo.setTitle('CONDUCTAS CONTRARIAS A LAS NORMAS DE CONVIVENCIA');
32 item.setTitle('Perturbar el normal desarrollo de las actividades de la clase');
33 item.setChoices([
34     item.createChoice('Dando voces.'),
35     item.createChoice('Levantandose sin motivo.'),
36     item.createChoice('Comiendo y/o bebiendo.'),
37     item.createChoice('Usando, manejando o exhibiendo aparatos ajenos a la
38     actividad del aula.')
39 ]);
40
41 //Establece nuevas opciones marcables para el campo item2
42 item2.setChoices([
43     item2.createChoice('No mantener la compostura adecuada al lugar y a la
44     actividad que se desarrolla en ella.'),
45     item2.createChoice('No colaborar sistematicamente en la realizacion de las
46     actividades orientadas al desarrollo del curriculum'),
47     item2.createChoice('No seguir las orientaciones del profesorado respecto a su
48     aprendizaje.'),
49     item2.createChoice('No traer al centro el material necesario para realizar
50     las actividades de clase.'),
51     item2.createChoice('Impedir o dificultar el ejercicio del derecho o el
52     cumplimiento del deber de estudiar de sus companeros.'),
53     item2.createChoice('No acudir con puntualidad a clase.'),
54     item2.createChoice('Faltar injustificadamente a clase.')
55 ]);
56
57 //Establece el titulo del campo item3 y las opciones marcables
58 item3.setTitle('Mostrar incorreccion y desconsideracion hacia');
59 item3.setChoices([
60     item3.createChoice('El profesorado.'),
61     item3.createChoice('Otros miembros de la comunidad educativa.')
62 ]);
63
64 //Establece el titulo del campo item4 y las opciones marcables
65 item4.setTitle('Causar pequenos danos en...');
66 item4.setChoices([
67     item4.createChoice('Las instalaciones.'),
68     item4.createChoice('Recursos materiales.'),
69     item4.createChoice('Documentos del centro.'),
70     item4.createChoice('En las pertenencias de los demas miembros de la comunidad
71     educativa.')
72 ]);
73
74 //Establece el titulo del campo item5 y las opciones marcables
75 item5.setChoices([
76     item5.createChoice('No colaborar en la limpieza y conservacion del
77     Instituto utilizando adecuadamente los aseos, papeleras, etc.')
78 ]);
79
80 //Tratamiento de la excepcion
81 } catch (err) {
82     //lanza la excepcion y el error lo muestra por consola
83     console.log('Failed with error %s', err.message);
84 }
85
86 //Generacion de la URL del formulario creado
87 Logger.log('Published URL: ' + form.getPublishedUrl());
88 Logger.log('Editor URL: ' + form.getEditUrl());
89 //Creacion del trigger que se ejecutara cuando se confirme el formulario.
90 ScriptApp.newTrigger("onFormSubmit").forForm(form).onFormSubmit().create();
91
92 return form.getPublishedUrl();
93 }
94
95 function onFormSubmit(e) {
96
97     // Obtiene la respuesta que se envio.
98     var formResponse = e.response;
99
100     // Obtiene la respuesta de cada uno de los campos del formulario
```




```
91     var itemResponses = formResponse.getItemResponses();
92
93     // La variable emailBody almacenara las respuestas que iran en
94     //el correo
95     var emailBody = "Formulario de incidencia:\n\n";
96
97     // Reune preguntas y respuestas de la variable emailBody
98     itemResponses.forEach(function(itemResponse) {
99         var title = itemResponse.getItem().getTitle();
100         var response = itemResponse.getResponse();
101         emailBody += title + "\n" + response + "\n\n";
102     });
103
104     //Llama a la funcion sendEmail pasando como variable las
105     //respuestas del formulario
106     sendEmail(emailBody);
107 }
108
109 //Funcion que envia los emails a las personas indicadas.
110 //getResponseEmail() devuelve la direccion de correo de la persona que envio una
    respuesta.
111 //si la configuracion de Form.setCollectEmail(collect) esta habilitada.
112 function sendEmail(emailBody) {
113     MailApp.sendEmail("jefaturadiurno@iesfernandoaguilar.es", "Nuevo formulario de
        conducta", emailBody);
114     MailApp.sendEmail(getRespondentEmail(), "Nuevo formulario de conducta", emailBody
        );
115 }
116 }
```

listaAlumno.gs

En este archivo podemos encontrar una única función que es

`listaAlumno()`

El cometido de esta función es listar los alumnos del instituto. Para ello se inicializa un vector de alumnos vacío y las variables `pageToken` y `page` para paginar los usuarios del dominio y poder mostrar más de quinientos usuarios.

Con un bucle **do while**, vamos iterando en cada una de las páginas y vamos llamando a la función

`AdminDirectory.Users.list(optionalArgs)`

Esta función llama al método `Users.list` que permite listar los usuarios del dominio y le pasamos una serie de parámetros opcionales:

- **Customer**. Permite decidir los clientes del dominio que queremos mostrar. Si usamos **"my_customer"** estamos usando nuestro ID único del dominio, es decir, indicamos que el dominio del cual queremos listar se corresponde con el dominio al cual pertenece nuestra ID de cliente única. Por tanto como somos usuarios del dominio del instituto, los clientes que van a listar se corresponderán también con los del dominio del instituto.
- **maxResults**. Establece la cantidad máxima de resultados que se mostrarán en cada iteración del bucle, continuando por donde se quedó.
- **pageToken**. Token para especificar la página siguiente en la lista.
- **orderBy**. Permite ordenar los resultados por el campo que nosotros elijamos que en nuestro caso es por el apellido de los usuarios del dominio.

Todos estos parámetros opcionales se pasan y se generará la lista de usuarios atendiendo a dichos parámetros. Luego con el bucle **for** comprobamos si el usuario es alumno con el correo habilitado al exterior o no y si tiene el campo del usuario **"organizations"** distinto a nulo.

De cumplirse dichas condiciones el alumno es introducido en el vector de alumnos mostrándose su apellido y su nombre.



Al final se ordena dicho vector para que los alumnos estén ordenados alfabéticamente y se muestren correctamente.

```
1  /**
2   * Funcion que lista los usuarios de un dominio
3   */
4  function listaAlumno() {
5
6      var persona = [];
7      var pageToken, page;
8
9      do {
10
11          //argumentos opcionales que se pasan a Users.list
12          var optionalArgs = {"customer":"my_customer", maxResults: 100, pageToken:
13              pageToken,
14              orderBy: 'familyName'}
15
16          //Se pagina los resultados obtenidos del dominio de Google.
17          page = AdminDirectory.Users.list(optionalArgs)
18          var allUsers = page.users
19
20          //Se recorre todos los usuarios y si pertenecen a alumnado o
21          //alumnado_email_abierto_al_exterior se anade a la lista de alumnos
22          for (i=0; i<allUsers.length; i++){
23
24              if (allUsers[i].organizations != null && (allUsers[i].orgUnitPath == "/"
25                  alumnado" || allUsers[i].orgUnitPath == "/alumnado/
26                  alumnado_email_abierto_al_exterior"))){
27
28                  persona.push(allUsers[i].name.familyName + ' ' + allUsers[i].name.givenName
29                      );
30
31              };
32          }
33          //Se pagina
34          pageToken = page.nextPageToken;
35
36      }while(pageToken);
37
38      //Se devuelve la lista de alumnos ordenados por apellido
39      return persona.sort((a, b) => a.localeCompare(b));
40  }
```

listaCurso.gs

La función

doGet()

es necesaria para crear una aplicación web con el servicio HTML para indicar al trozo de código cómo debe entregar la página. Sin ella la aplicación web no es capaz de llamar a las funciones JavaScript ni se ejecutaría el sistema.

La función

listarCursos()

recorre todos y cada uno de los alumnos de la misma forma que lo hace listaAlumno(), pero en este caso queremos acceder al campo de *Departamento*, no al nombre del alumno. Posteriormente el curso se guarda en un vector. Tras listar el curso de cada alumno tenemos muchos duplicados y están desordenados los cursos. Es por ello que haciendo uso del método

cursos.filter()

eliminamos los duplicados y nos quedamos con un vector de elementos únicos, en definitiva serían todos los cursos del instituto sin repetir ninguno.



Aún siguen desordenados, es por ello que con el método

```
listCur.sort()
```

ordenamos los cursos para que el desplegable de cursos sea presentable y legible.

```
1 /**La funcion doGet() permite implementar el codigo como una
2  * aplicacion web.
3  */
4
5 function doGet() {
6     return HtmlService.createHtmlOutputFromFile('index').setTitle('Formulario de
7     incidencias').setFaviconUrl('https://cdn.icon-icons.com/icons2/3023/PNG/512/
8     notebook_address_book_book_icon_188753.png');
9 }
10
11 /** Lista los cursos de los que dispone el instituto */
12 function listarCursos() {
13
14     var cursos = [];
15     var listCur = [];
16     var pageToken, page;
17
18     do {
19
20         //Parametros opcionales que se pasan a User.list
21         var optionalArgs = {"customer": "my_customer", maxResults: 100, pageToken:
22         pageToken, orderBy: 'familyName'}
23         page = AdminDirectory.Users.list(optionalArgs)
24         var allUsers = page.users
25
26         //Recorre todos los alumnos que si son alumnado o
27         alumnado_email_abierto_al_exterior
28         //saca el curso de cada alumno y lo guarda en un vector
29         for (i=0; i<allUsers.length; i++){
30             userDetails = allUsers[i]
31             if (allUsers[i].organizations != null && allUsers[i].orgUnitPath == "/"
32             alumnado" || allUsers[i].orgUnitPath == "/alumnado/
33             alumnado_email_abierto_al_exterior"){
34                 cursos.push('${allUsers[i].organizations[0].department}');
35
36             //Elimina los cursos duplicados
37             listCur = cursos.filter((item,index)=>{
38                 return cursos.indexOf(item) === index;})
39
40         }
41
42         pageToken = page.nextPageToken;
43     }while(pageToken);
44
45     //Ordena los cursos y lo devuelve.
46     listCur.sort();
47     console.log(listCur);
48
49     return listCur;
50 }
```

index.html

Todos los archivos anteriores contienen las funciones necesarias para generar el formulario, listar alumnos, sacar los cursos etc... Para poder hacer que el desplegable de cursos modifique o actualice el desplegable de alumnos es necesario la creación de una sencilla aplicación Web, es por ello necesario la existencia de este archivo index.html

En él se encuentra los diferentes campos que van a mostrar en la página y una pequeña sección de script que tiene como finalidad:

- Generar el desplegable de cursos para que el usuario sea capaz de elegirlo.
- Los EventListener que detectarán cambios en el valor del desplegable o que detecte que se le da al botón de confirmar y salte el evento para llamar a la función que genera el parte



de incidencia, pasándole como valor el contenido del desplegable cursos. De esta forma el desplegable de alumnos se generará haciendo uso de la función listaAlumno() pasándole el curso desde el index.html y generará el formulario con el desplegable de alumnos deseado.

Para hacer posible el uso de la aplicación web necesitamos un fragmento de código html con el que poder presentar la página y llamar a las funciones .gs de nuestro proyecto (para generar el desplegable de cursos y crear el formulario). Esta parte del proyecto la página web no cuenta con ningún estilo.

```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <base target="_top">
5     <title>Formulario incidencia</title>
6   </head>
7   <body>
8     <div class='container'>
9       <h1>Bienvenido al formulario de incidencias</h1>
10
11     <script>
12
13       var listado;
14
15       function onSuccess(elcurso) {
16
17         listado = document.getElementById('lista');
18         listado.innerHTML = elcurso;
19         listado.addEventListener('change', (e) =>{
20           console.log(e.target.value);
21
22           function creaFormulario(creacion) {
23
24             var cForm = document.getElementById('formu');
25             //cForm.innerHTML = 'Formulario creado: ' + '<a href="' + creacion + '>'>
26             Enlace al formulario</a>';
27             //cForm.innerHTML = 'Formulario creado: ' + '<a href="'+creacion+'>'>
28             enlace</a>';
29             window.location.replace(creacion);
30
31           }
32
33           google.script.run.withSuccessHandler(creaFormulario).parteIncidencia(e.
34             target.value);}
35
36           google.script.run.withSuccessHandler(onSuccess).cargarCursos();
37         </script>
38         <p>Lista de cursos.</p>
39         <select id="lista"></select>
40         <p>Seleccione curso y espere a ser redirigido al formulario</p>
41         <div id='formu'></div>
42       </div>
43
44     </body>
45 </html>
```

Ya se han listado y detallado los archivos de la primera parte del proyecto. Como se comentó anteriormente, esa parte corresponde a la solución que se pensaba en un principio que era crear un formulario desde cero ya fuese con el desplegable de cursos dentro del formulario o fuera en la página web. Sin embargo el crear un formulario nuevo cada vez hacía muy complicado el seguimiento de los partes creados para cada uno de los alumnos.

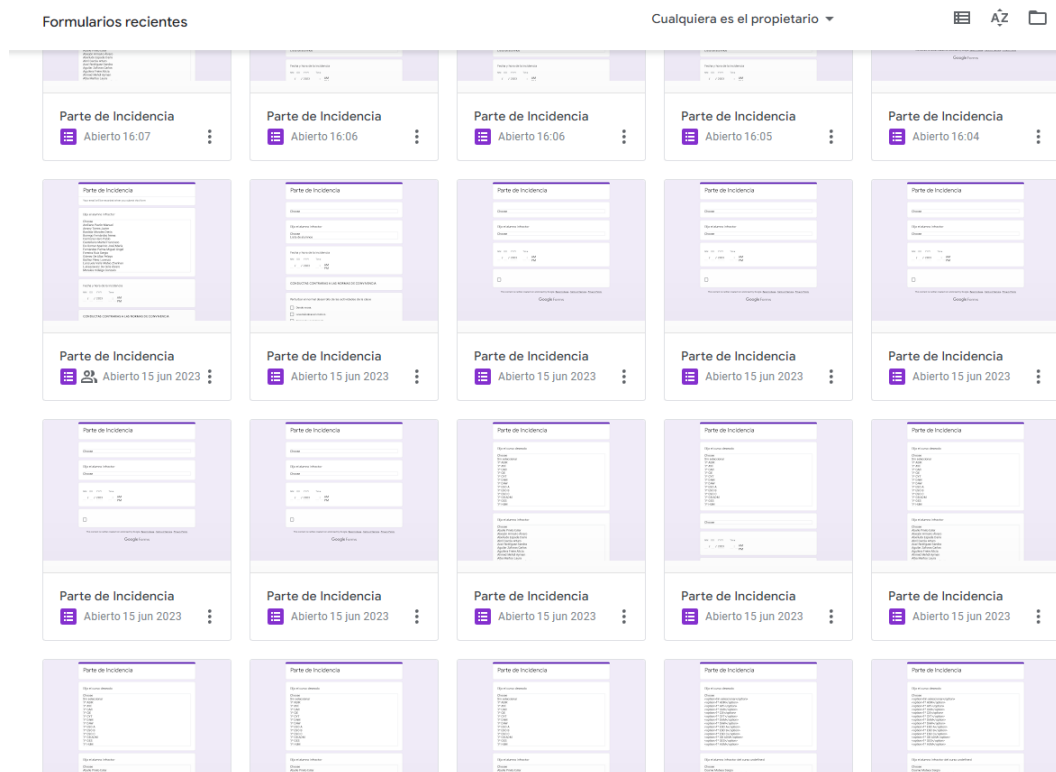


Figura 4: Ejemplo de la numerosa cantidad de formularios que se crean.

En la figura 4 se puede observar una pequeña muestra de la cantidad de formularios que se pueden crear. Si con el tiempo queremos volver a mirar un formulario en concreto para su revisión puede ser muy tedioso llegar a él además que la vinculación a la Hoja de cálculo para el volcado de la información también puede resultar farragoso.

Por este motivo se dio una pequeña vuelta al tema. No se va a crear un formulario nuevo cada vez, si no que vamos a partir de un formulario ya creado, ya sea desde cero por código .gs con Google Apps Script o creado con la interfaz gráfica de Google Forms. A partir de ese formulario ya creado se van a ejecutar los diferentes fragmentos de código para realizar la generación del desplegable de cursos, la actualización del desplegable de alumnos...

2.1.2. Segunda parte de la ejecución.

En esta segunda parte del proyecto a diferencia de la primera parte, tenemos un contenedor, que es el Formulario de Google del cual parte el proyecto. En la primera parte no se partía de ningún formulario ya creado si no que se creaban siempre de cero.

Datos Del Proyecto



Apps Script

Contenedor
Parte de Incidencia

Propietario
Yo

Figura 5: Proyecto que tiene un contenedor.

Para hacer que el proyecto tenga el formulario como contenedor tenemos que darle a la opción de "Editor de secuencias de comandos" mostrado en la figura de justo abajo.

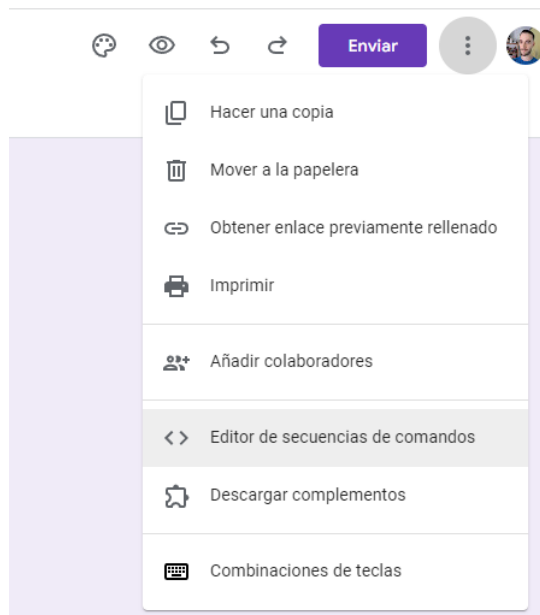


Figura 6: Creación de un proyecto con un contenedor.

Procedemos a listar los archivos usados en esta segunda parte del proyecto.

- index.html
- actualizaUsuario.gs
- buscaUsuario.gs
- guardaCurso.gs
- cargaCurso.gs

index.html

En este segundo archivo index.html tenemos una página web parecida que tiene la misma mecánica de generar el desplegable de cursos pero en esta ocasión la página llama a una función que abre un formulario creado anteriormente en vez de llamar a la función que crea un formulario cada vez e igualmente le pasa el valor elegido en el desplegable de cursos a la función de

`actualizarAlumnos()`

Igualmente la carga de cursos también difiere a la primera parte del proyecto y lo detallaremos más adelante.

```
1      <!DOCTYPE html>
2      <html>
3      <head>
4          <base target="_top">
5          <title>Formulario incidencia</title>
6      </head>
7      <body>
8          <div class='container'>
9              <h1>Bienvenido al formulario de incidencias</h1>
10
11          <script>
12
13              var listado;
14
15              function onSuccess(elcurso) {
16
17                  listado = document.getElementById('lista');
18                  listado.innerHTML = elcurso;
19                  listado.addEventListener('change', (e) =>{
20                      console.log(e.target.value);
```



```
21
22     function creaFormulario(creacion) {
23
24         //var cForm = document.getElementById('formu');
25         //cForm.innerHTML = 'Formulario creado: ' + '<a href="' + creacion + '>
Enlace al formulario</a>';
26         //cForm.innerHTML = 'Formulario creado: ' + '<a href="'+creacion+'>
enlace</a>';
27         window.location.replace("https://docs.google.com/forms/d/e/1
FAIpQLSfEwAM0-5J3oU2saTysynllsIPrLCyTtZEgQ_b3qmNuuRjwQ/viewform");
28
29     }
30
31     google.script.run.withSuccessHandler(creaFormulario).actualizarAlumnos(e.
target.value);})
32
33 }
34
35 google.script.run.withSuccessHandler(onSuccess).cargarCursos();
36 </script>
37 <p>Lista de cursos.</p>
38 <select id="lista"></select>
39 <p>Seleccione curso y espere a ser redirigido al formulario</p>
40 <!--<div id='formu'></div-->
41 </div>
42
43 </body>
44 </html>
```

actualizaUsuario.gs

En este archivo nos encontramos con dos funciones. La primera función es

obtenerID ()

No es una función muy importante para la ejecución del sistema pero sí es útil para comprender y sacar ciertos datos con los que poder trabajar. Si vemos su interior podemos ver en la línea dos la función

FormApp.openById().getItems().getId();

que devuelve un objeto de tipo entero que lo guardamos en la variable **item**. Eso devuelve el ID del Item 0 (que en nuestro formulario se corresponde con el desplegable de alumnos) y que ese campo se encuentra en el formulario existente el cual indicamos pasando su ID.

El ID de un formulario de Google es una parte de su URL:

https://docs.google.com/forms/d/1ZnWbl4OGsL3rcsQlZ_zlcHqGSz8pZbVUaUH7I_YcxX0/edit

siendo la parte marcada en negrita de la URL el ID de dicho formulario. De esta forma indicamos a la función cual es el formulario al que nos referimos, el item o campo de ese formulario y lo que queremos hacer con ese campo, como obtener su id, su título etc...

```
1 function obtenerID() {
2     var item = FormApp.openById('1M-DbMC8giF0JwRYdRXqzL-smCTGCBcyRhouizBTZt00').
getItems()[0].getId();
3     console.log(item);
4     //843446016
5 }
```

El número comentado es el ID del campo del desplegable. Podríamos directamente pasarle el valor que devuelve el método **getItems()[0].getId()** en vez de escribir el número y luego escribirlo manualmente. Esto se hizo así en el momento para comprobar que realmente devuelve el ID de forma correcta y que pasando dicho número el desplegable de alumnos se veía alterado.

Posterior a la función **obtenerID()**, tenemos la función

actualizarAlumnos(curso)



que se encarga de abrir también el correspondiente formulario pasando su ID y lo guardamos en la variable form. Luego en la variable item guardaremos el tipo de campo del formulario llamando a los métodos de la siguiente forma:

```
form.getItemById(form.getItems()[0].getId()).asListItem();
```

Lo que hace exactamente es obtener el item o campo que queremos modificar con el método **getItemById()** mediante el ID que sacamos llamando en el interior de este método a otro método **form.getItems()[0].getId()**. Posteriormente indicamos que ese item es de tipo "lista desplegable" tal y como devuelve el método **asListItem()**. Por tanto en la variable item tenemos ahora mismo guardado el tipo del campo de desplegable de alumnos.

Justo después tenemos una variable options que llama a la función **buscaUsuario()**, función a la que le pasamos el curso seleccionado en el desplegable del index.html que guardará todos los usuarios correspondientes al curso pasado por valor, para posteriormente llamar a la función **.setChoiceValues()** que creará en el campo item el desplegable con los valores de options.

```
1 function actualizarAlumnos(curso) {  
2     var form = FormApp.openById('1M-DbMC8giFOJwRYdRXQzL-sMCTGCBcyRhouizBTZt00');  
3     var item = form.getItemById(form.getItems()[0].getId()).asListItem();  
4     var options = buscarUsuario(curso);  
5     item.setChoiceValues(options);  
6  
7 }
```

buscaUsuario.gs

En este archivo podemos encontrar una única función que es

```
buscarUsuario()
```

función a la que se le pasa el curso seleccionado en el desplegable de la página web.

Creamos un vector vacío de alumnos y como vimos anteriormente llamamos a la función **AdminDirectory.Users.list(optionalArgs)** que le pasamos una serie de parámetros opcionales que son:

- **Customer**. Los clientes del dominio al cual nos queremos referir. Usando **"my_customer"** podemos referirnos a los usuarios de nuestro dominio.
- **maxResults**. El máximo de resultados que va a devolver, en este caso van a ser cincuenta.
- **orderBy** indica que orden debe seguir los resultados mostrados.
- **query**. Permite indicar ciertos filtrados o sacar cierta información. En nuestro caso filtra los alumnos por el siguiente criterio: los que pertenezcan al Departamento seleccionado, es decir que pertenezcan al curso que pasamos con el desplegable de cursos.

Si por el motivo que fuese ningún usuario cumple los requisitos indicados en **optionalArgs** se cumplirá la condición del **if** y mostrará un mensaje por consola.

Posteriormente de haber usuarios, con el bucle **for** se recorren todos los usuarios que sí cumplen con los requisitos, se eliminan del filtrado los que tengan el campo **"organizations"** nulo y se obtienen los apellidos y nombre de los no nulos y se guardan en el vector de alumnos que posteriormente será ordenado para que aparezcan en orden alfabético respetando hasta los apellidos con tilde y no se traten como diferentes.

```
1 function buscarUsuario(curso) {  
2     var alumnos = [];  
3     const optionalArgs = {  
4         "customer": "my_customer",  
5         "maxResults": 50,  
6         "orderBy": "familyName",  
7         "query": "orgDepartment='"+curso+"'";  
8  
9     };  
10    try {
```



```
11
12     const response = AdminDirectory.Users.list(optionalArgs);
13     const users = response.users;
14     if (!users || users.length === 0) {
15         console.log('No se encontro ningun usuario.');
```

return;

```
17     }
18     console.log('Users:');
19     for (const user of users) {
20         if (user.organizations != null){
21             alumnos.push(user.name.familyName + " " + user.name.givenName)
22             //console.log(user);
23             //console.log('%s (%s)', user.primaryEmail, user.name.fullName, user.
24             organizations[0].department);
25         }
26     }
27 } catch (err) {
28     console.log('Failed with error %s', err.message);
29 }
30 return alumnos.sort((a, b) => a.localeCompare(b));;
```

guardaCurso.gs

Dentro de este archivo tenemos la función

guardarCursos()

El cometido de esta función es leer el curso de todos los usuarios del instituto para luego guardarlos en un vector, eliminar los duplicados y ordenarlos. Una vez hecho todo esto tenemos el vector de cursos del colegio que volcaremos en una hoja de cálculo.

Para saber a qué hoja de cálculo nos referimos la abrimos con

SpreadsheetApp.openByUrl('URL de la hoja').getSheets()[0];

Guardamos en hojaCurso el tipo de objeto que devuelve la función anterior y con el uso de **getRange()** vamos volcando en la hoja de cálculo fila por fila con un bucle for los datos del vector de cursos que se ha rellenado de la misma forma que ya hemos comentado en un fragmento de código anterior.

En el momento en el que haya un curso nuevo o se elimine un curso ya existente, al ejecutar este script se actualiza la hoja de cálculo con los cursos que imparte actualmente el colegio, pues si se cumple la condición del if, es decir, existe alguna fila con datos, clearContent() limpia todas las filas para posteriormente volver a escribir en ellas, permitiendo que la lista de cursos quede totalmente actualizada.

```
1 function guardarCursos() {
2
3     var hojaCurso = SpreadsheetApp.openByUrl('https://docs.google.com/spreadsheets/d
4     /1DzAGaUbqoCilc-vOB_gKWih_vzgupydtFGDcKvjuRmg/edit#gid=0').getSheets()[0];
5     var cursos = [];
6     var listCur = [];
7     var pageToken, page;
8
9     do {
10
11         var optionalArgs = {"customer": "my_customer", maxResults: 100, pageToken:
12         pageToken, orderBy: 'familyName'}
13         page = AdminDirectory.Users.list(optionalArgs)
14         var allUsers = page.users
15
16         //recorre los alumnos del instituto y se queda con el campo del Departamento
17         (curso)
18         for (i=0; i<allUsers.length; i++){
19
20             userDetails = allUsers[i]
21
22             if (allUsers[i].organizations != null && allUsers[i].orgUnitPath == "/"
23             alumnado" || allUsers[i].orgUnitPath == "/alumnado/
24             alumnado_email_abierto_al_exterior"){
```




```
20     cursos.push(`${allUsers[i].organizations[0].department}`);
21   };
22
23   listCur = cursos.filter((item,index)=>{
24     return cursos.indexOf(item) === index;
25   })
26
27   }
28
29   pageToken = page.nextPageToken;
30
31   }while(pageToken);
32
33   listCur.sort();
34   listCur.unshift('Sin seleccionar');
35
36   //limpia la hoja de calculo si hay alguna fila escrita
37   if (hojaCurso.getLastRow() >=1){
38     hojaCurso.getRange(1,1,hojaCurso.getLastRow(),1).clearContent();
39   }
40
41   //escribe en la hoja de calculo los valores del vector de cursos
42   for(var i=1;i<=listCur.length;i++){
43     hojaCurso.getRange(i, 1).setValue(listCur[i-1]);}
44
45 }
```

cargaCurso.gs

En este archivo podemos encontrar la función

cargarCursos()

Esta función se encarga de leer la hoja de cálculo donde hemos guardado anteriormente los cursos del instituto y pasárselos a la página web para crear el desplegable de cursos.

Para abrir la hoja se procede de la misma forma que con guardarCursos(), se llama a la función SpreadsheetApp.openByUrl() le pasamos la URL de la hoja de cálculo que queremos leer y guardamos el tipo de objeto en hojaCurso.

Con el bucle for vamos leyendo cada una de las filas y vamos metiendo el valor en el vector de cursos concatenando con las etiquetas `< option >` de HTML para que el desplegable se cargue correctamente en la página web.

La idea de hacerlo así es mejorar la velocidad a la que se genera dicho desplegable, pues leer de una hoja de cálculo con menos de 100 campos es más rápido que leer todos los alumnos y sacar de cada uno de ellos el curso y hacer los pertinentes arreglos para que quede el vector de cursos como se desea, sin duplicados y ordenado.

```
1 function cargarCursos() {
2
3   const lCurso = [];
4
5   //abre la hoja de calculo y recorre la primera columna hasta el final para
6   //leer los cursos y pasárselos al desplegable de cursos de la pagina web
7   var hojaCurso = SpreadsheetApp.openByUrl('https://docs.google.com/spreadsheets/d
8   /1DzAGaUbqoCilc-vOB_gKWih_vzgupydtFGDcKvjuRmg/edit#gid=0').getSheets()[0];
9   for(var i=1;i<=hojaCurso.getLastRow();i++){
10     lCurso.push('<option>'+hojaCurso.getRange(i, 1).getValue()+'</option>');}
11     console.log(lCurso);
12   return lCurso;
13 }
```



actualizaProfesores.gs

Podemos encontrar dos funciones.

leerProfesores()

Hace una búsqueda en el dominio con AdminDirectory.Users.list() con los parámetros con los que ya estamos familiarizados para listar todos los profesores.

```
1 //Actualiza la lista de profesores del formulario.
2 function leerProfesores() {
3
4     var listProf = [];
5     var pageToken, page;
6
7     do {
8
9         var optionalArgs = {"customer": "my_customer", maxResults: 300, pageToken:
10         pageToken, orderBy: 'familyName'}
11         page = AdminDirectory.Users.list(optionalArgs)
12         var allUsers = page.users
13
14         //recorre los profesores del instituto
15         for (i=0; i<allUsers.length; i++){
16
17             userDetails = allUsers[i]
18
19             //Se comprueba que sea profesor y no muestre los profesores de prueba
20             if (allUsers[i].orgUnitPath == "/profesorado" && !allUsers[i].name.
21             givenName.includes('profesor')){
22                 listProf.push(allUsers[i].name.familyName + ' ' + allUsers[i].name.
23                 givenName);
24             };
25         }
26
27         pageToken = page.nextPageToken;
28     }while(pageToken);
29
30     return listProf.sort((a, b) => a.localeCompare(b));;
```

actualizarProfesores()

La idea de esta función es ejecutarla cuando los profesores del instituto cambien para refrescar el desplegable de profesores y sea actual a la situación del instituto.

```
1 function actualizarProfesores() {
2 var form = FormApp.openById('1-vBCzam5-rh2FugOpyqukk5mq9cY2GhJaHwMCwUaPck');
3 var item = form.getItemById(form.getItems()[1].getId()).asListItem();
4 item.setTitle('Apellidos y nombre del/la profesor/a denunciante')
5 item.setChoiceValues(leerProfesores());
6
7 }
```

triggerEmail.gs

Este archivo contienen 3 funciones importantes para habilitar el envío de correos al rellenar un formulario.

La primera función es

crearTrigger()

Su cometido es crear el trigger en caso de no existir, si existe no hace nada. Sin el trigger es imposible que salte la ejecución del envío de emails. Los triggers pueden verse en el proyecto a la izquierda haciendo clic en el icono del reloj.



```
1 function crearTrigger() {  
2  
3   var form = FormApp.openById('1-vBCzam5-rh2Fug0pyqukk5mq9cY2GhJaHwMCwUaPck');  
4   var numTriggers = ScriptApp.getProjectTriggers();  
5   if(numTriggers.length > 0)  
6     return;  
7  
8   ScriptApp.newTrigger("onFormSubmit").forForm(form).onFormSubmit().create();  
9 }
```

En cuanto a la función

onFormSubmit(e)

Recorre cada uno de los campos del formulario para guardar el título y la respuesta de cada uno de ellos y almacenarlos en emailBody que contendrá en esencia, las respuestas del formulario para mandarlo por correo. También tiene una variable emisor que recoge un vector de correos de personas que han respondido el formulario, del cual nos interesa la última posición que es la última persona que ha respondido el formulario.

```
1 function onFormSubmit(e) {  
2  
3   var form = FormApp.openById('1-vBCzam5-rh2Fug0pyqukk5mq9cY2GhJaHwMCwUaPck');  
4   var emisor = form.getResponses();  
5   var formResponse = e.response;  
6   var itemResponses = formResponse.getItemResponses();  
7   var emailBody = "Respuestas del formulario rellenado:\n\n";  
8  
9   itemResponses.forEach(function(itemResponse) {  
10     var title = itemResponse.getItem().getTitle();  
11     var response = itemResponse.getResponse();  
12     emailBody += title + "\n" + response + "\n\n";  
13   });  
14   //emailBody += formResponse.getItemResponses()[2].getResponse() + " EL CURSO";  
15   //emailBody += buscarProfesor(formResponse.getItemResponses()[2].getResponse()) +  
16     " CORREO DEL TUTOR";  
17   sendEmail(emailBody, emisor[emisor.length-1].getRespondentEmail(), buscarProfesor  
18     (formResponse.getItemResponses()[2].getResponse()).toString());  
19 }
```

La función

sendEmail(emailBody, emisor, tutor)

Manda un email con las respuestas del formulario a tres personas, al tutor del curso, a quien envió el formulario y a jefatura diurna.

```
1 function sendEmail(emailBody, emisor, tutor) {  
2  
3   MailApp.sendEmail(tutor, "Nueva respuesta de formulario", emailBody);  
4   MailApp.sendEmail(emisor, "Nueva respuesta de formulario", emailBody);  
5   MailApp.sendEmail("CORREO DE JEFATURA", "Nueva respuesta de formulario",  
6     emailBody );  
7 }
```

Para poder hacer posible la recolección de los correos de las personas que han rellenado y enviado el formulario es necesario que el atributo de la variable Collect Email sea true. Para esto existe la función

establecerCollect()

Su ejecución establece el valor en true y devuelve dicho valor por consola para confirmar que está todo en orden.

```
1 function establecerCollect() {  
2  
3   var form = FormApp.openById('1-vBCzam5-rh2Fug0pyqukk5mq9cY2GhJaHwMCwUaPck');  
4   form.setCollectEmail(true);  
5   const collect = form.collectsEmail();  
6   console.log(collect);  
7 }
```



```
7   const formResponses = form.getResponses();
8   console.log(formResponses[formResponses.length-1].getRespondentEmail());
9
10
11 }
```

buscaProfesor.gs

Dentro de este archivo tenemos una función llamada

`buscarProfesor(curso)`

Su función es devolver el profesor que es tutor del curso que pasamos por el desplegable del index.html. Esta necesidad nace de mandar un correo al profesor con las respuestas del formulario del alumno del cual es tutor.

```
1 function buscarProfesor(curso) {
2   var profesores = [];
3   const optionalArgs = {
4     "customer": "my_customer",
5     "maxResults": 50,
6     "orderBy": "familyName",
7     "query": "orgTitle="+""+curso+" "+""
8   };
9   try {
10
11     const response = AdminDirectory.Users.list(optionalArgs);
12     const users = response.users;
13     if (!users || users.length === 0) {
14       console.log('No se encontro ningun usuario.');
```



2.2. Configuraciones.

Una vez hablado de la estructura del proyecto, el cómo se organiza los archivos y explicado el contenido de cada uno de ellos podemos pasar a hablar de la configuración que se debe hacer del proyecto para que funcione sin problema.

Para poder hacer uso de `AdminDirectory.Users.list()` es necesario en el proyecto activar el servicio de AdminDirectory.

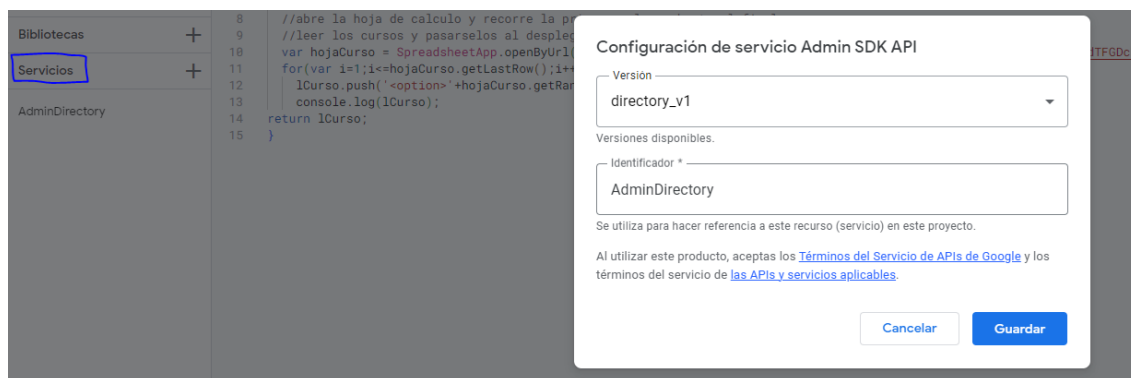


Figura 7: Habilitando el servicio de Admin SDK API.

Los permisos de cada una de las hojas de cálculo y formulario deben ser revisadas minuciosamente. En caso de no disponer de los permisos necesarios la aplicación web no listará el desplegable de cursos o no se mostrará el contenido de las hojas de cálculo ni se podría rellenar el formulario. Es conveniente pues que se permita permiso de lectura al menos a todos los usuarios del instituto.

A fin de cuentas cuando se utiliza la página web se hace uso de cuenta de Google, automáticamente de forma interna Google comprueba los permisos y el desplegable de cursos depende de una hoja de cálculo. Si no tienes permiso de lectura de esa hoja de cálculo dicho desplegable nunca se mostrará impidiendo el uso normal del sistema.

En el ejemplo que vamos a mostrar se ve que le damos permisos de lectura a todo el instituto y si acaso Diego es el encargado de editarlo. Este permiso pienso que es necesario para que Diego pueda hacer uso del script `guardaCursos()` y se escriba de forma correcta y no haya problemas de ejecución. A fin de cuentas el permiso de Editor deberá tenerlo los administradores que se encarguen de mantener el sistema.

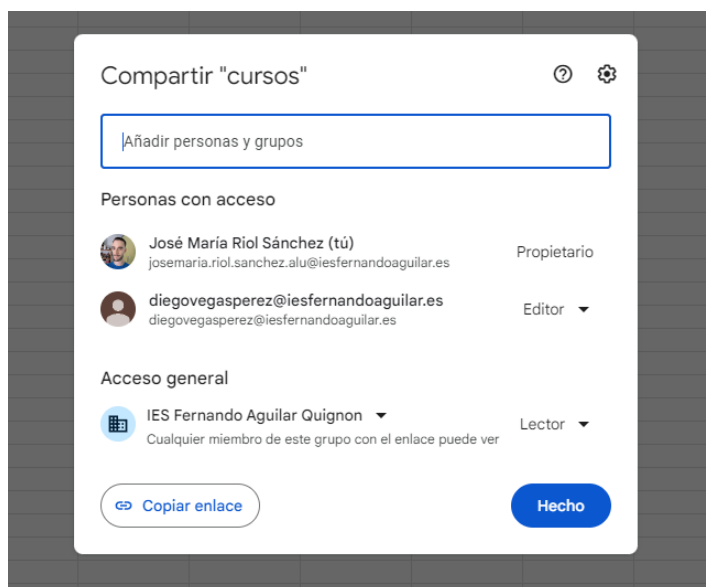


Figura 8: Permisos de la hoja de cálculo

Para que el trigger de los Emails funcione correctamente es necesario comprobar que esté activado correctamente, para ello...

+ Añade un filtro			
Propietario	Última ejecución	Implementación	Evento
Yo	22 jun 2023, 19:51:07	Principal	De un formulario - Al enviarse el formulario

Figura 9: Consultando la lista de triggers

En el caso de no estar activado el trigger **se insta** mirar el apartado de **"Manual de instalación"** para ver todo el proceso que es necesario realizar para habilitar los triggers y la recolección de los correos de aquellas personas que completan el formulario.

3. Fase de pruebas.

En cuanto a la batería de pruebas empezaremos a contar desde que se pudo realizar el primer formulario de Google creado con Google Apps Script, donde merece la pena echar el vistazo y puede resultar interesante. Obviaremos entonces las etapas anteriores a esta pues tenían más que ver con cómo se iba a desarrollar el sistema.

3.0.1. Primeras pruebas

Partimos de un formulario que contiene los dos desplegables (cursos y alumnos). Estos desplegables no satisfacen la necesidad del cliente pues los desplegables no eran dependientes uno del otro.



Figura 10: Desplegables dentro del formulario.

Es por ello que no nos quedamos con esta solución, se precisa una mejora en el sistema. Esa mejora consiste en encontrar la manera que el desplegable de alumnos dependa del curso seleccionado anteriormente. Por eso tras una minuciosa búsqueda se encontró una manera haciendo uso de una aplicación web que mediante eventos y listener se registrara el valor seleccionado del desplegable cursos y se enviara a otras funciones de nuestro programa.

3.0.2. Segundas pruebas

En esta ocasión tenemos creada una simple página web que nos permite elegir el curso y se nos crea un formulario con el desplegable de alumnos correspondiente.

Figura 11: Página web simple.

El problema en esta ocasión se observó en el tiempo que tardaba en cargar el desplegable de cursos que podía rondar alrededor del medio minuto, un tiempo de ejecución demasiado largo. Una vez seleccionado el curso de la lista había otro tiempo de espera demasiado largo, haciendo que la simple creación de un formulario llevase alrededor de 1 minuto. Esto se debía a que se hacía una búsqueda secuencial cada vez que se ejecutaba esos fragmentos de código para generar el desplegable de cursos y el formulario.

Para la mejora de dichos tiempos de ejecución se llevó a cabo ciertos cambios:

- En cuanto a la generación del formulario con el correspondiente desplegable de alumnos la solución es simple. El método `users.list` permite además de ordenar, controlar los resultados que se muestran e indicar el dominio al cual quieres hacer el listado de usuarios, con el parámetro "query" se puede filtrar por una serie de campos y uno de ellos es el departamento. Esto hace que internamente Google realice una búsqueda de usuarios que cumplan los requisitos esta-

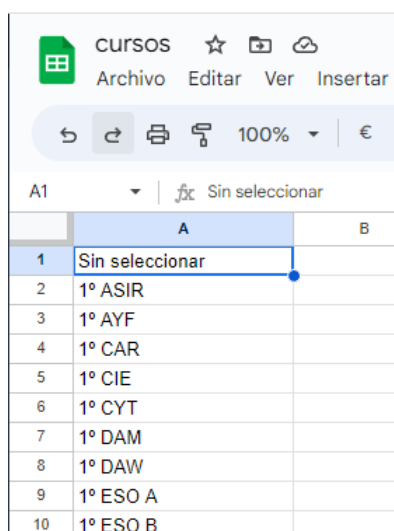


blecidos y los devuelve de forma más rápida que con el método anterior. Todos los parámetros de la consulta se pasan como parámetros opcionales a la función *AdminDirectory.Users.list()*.

```
function buscarUsuario(curso) {  
  var alumnos = [];  
  const optionalArgs = {  
    "customer": "my_customer",  
    "maxResults": 50,  
    "orderBy": "familyName",  
    "query": "orgDepartment="+""+curso+" "+""  
  };  
};
```

Figura 12: Filtrado de alumnos por el curso seleccionado.

- Por otro lado tenemos el desplegable de cursos. La solución que se llevó a cabo para mejorar la velocidad a la que se generaba el desplegable fue leer el listado de cursos de una hoja de cálculo. La lectura de cincuenta y pico filas de una única columna de una hoja de cálculo era mucho más rápida que la lectura del método anterior, mejorando bastante la generación del desplegable de cursos en la página web.



	A	B
1	Sin seleccionar	
2	1º ASIR	
3	1º AYF	
4	1º CAR	
5	1º CIE	
6	1º CYT	
7	1º DAM	
8	1º DAW	
9	1º ESO A	
10	1º ESO B	

Figura 13: Hoja de cálculo con los cursos del instituto.

3.0.3. Terceras pruebas

Con todo esto aún había un problema bastante importante que abordar. Tal y como está planteado el sistema ahora se van generando formularios nuevos cada vez. Esto impide el seguimiento y el rastreo rápido y cómodo de los partes de incidencia. A largo plazo podríamos tener más de mil partes de incidencia en Google Form y eso es totalmente inviable. Además de hacer tediosa la labor de vincular la hoja de cálculo al parte de incidencia con el objetivo de volcar la información en él.

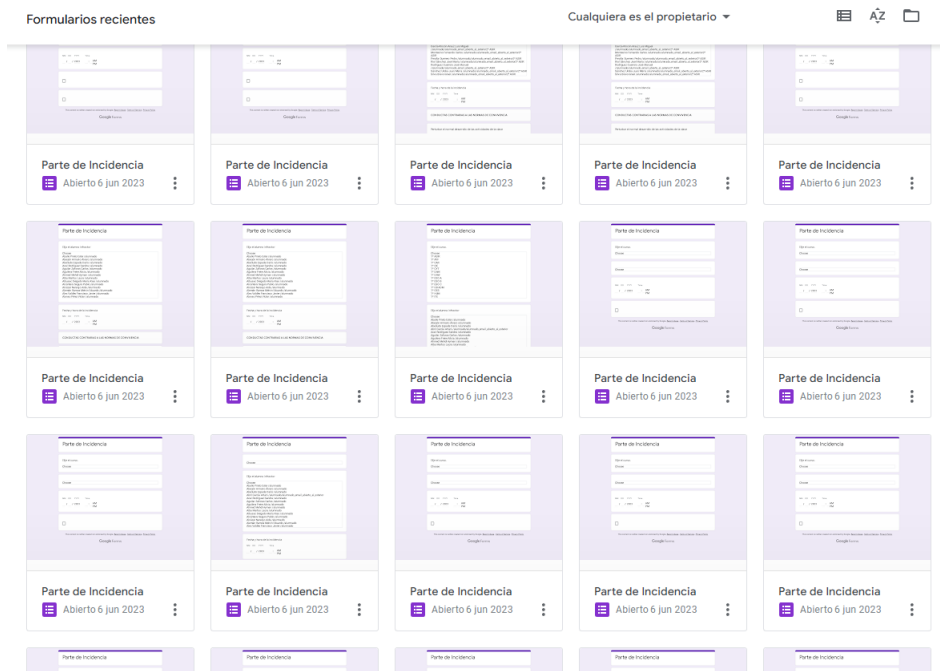


Figura 14: Generación de formularios descontrolada.

Para este problema se tuvo que dar un planteamiento algo distinto al proyecto, pues desde el principio se pensó que el sistema debía crear un nuevo formulario con los campos necesarios con el fin de poder generar el desplegable de alumnos acorde al curso elegido. Tras mirar exhaustivamente diferentes soluciones, había un planteamiento muy sencillo en el cual caí. En vez de hacer que el código genere formularios vamos a partir de un formulario ya creado y ese formulario ya creado, será el que varíe de una ejecución a otra. Esto soluciona el problema de la generación indiscriminada de formularios y además con tener ese formulario ya vinculado a las hojas de cálculo, todas las respuestas que se hagan en ella se podrán volcar en ella de forma totalmente cómoda.

4. Documentación del sistema.

4.1. Introducción a la aplicación.

Como ya se ha comentado anteriormente nuestra aplicación es una aplicación Web que nos sirve un desplegable de cursos. Este desplegable nos permite seleccionar el curso del alumno al cual hay que ponerle un parte de conducta y abrir un formulario con la lista de alumnos de dicho curso.

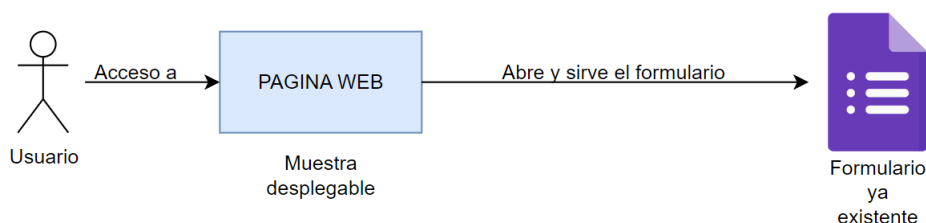


Figura 15: Esquema general del sistema.

Cuando accedemos a la página y seleccionemos el curso deseado se nos abrirá el formulario alojado en Google Forms con la lista de alumnos del curso elegido. Se rellenarán los campos del formulario y se procederá al envío.

Una vez enviado el formulario se procederá al envío de tres correos con el contenido del formulario. Un correo se enviará al profesor que rellenó el formulario, otro correo al tutor del curso elegido y



un tercer correo llegará a la cuenta de correo de jefatura diurno.

Por otro lado tenemos la capacidad de guardar en una hoja de cálculo la lista de cursos disponibles en el instituto con el objetivo de cargar los cursos de esta hoja de cálculo y acelerar el proceso de lectura para el desplegable de la página web. Esto permite que no se deban cambiar los cursos de forma manual, ahorrando trabajo a la persona encargada de mantener el sistema.

La página Web se sirve gracias a Google Sites al cual le hemos pasado la URL de la implementación de nuestro sistema (proyecto) para que lo ejecute sin estar alojado en nuestros Google Drive. Posteriormente como se mencionó hace un momento, el formulario que se abre se encuentra en Google Forms.

4.2. Manual de instalación.

El manual de instalación parte con el formulario de Google Form ya creado. Para la instalación del sistema hay que tener en cuenta los siguientes aspectos:

- Creación de un proyecto en Google Apps Script a partir del Formulario de partes de incidencia con todos los fragmentos de código necesarios y habilitar el servicio de AdminDirectory para su correcto funcionamiento. Además hay que prestar especial atención a los permisos que tengan las diferentes hojas de cálculo y el formulario.

Para crear dicho proyecto tendremos que irnos a la vista de edición de nuestro formulario y elegir el editor de secuencias de comandos.

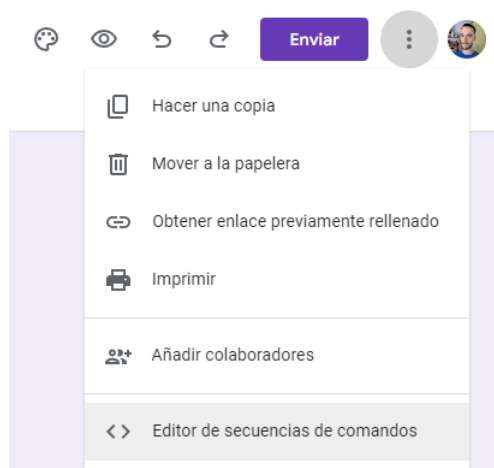


Figura 16: Crear proyecto a partir de un formulario.

Una vez creado el proyecto de esta forma, nos aparecerá en la lista de proyectos de Google Apps Script.

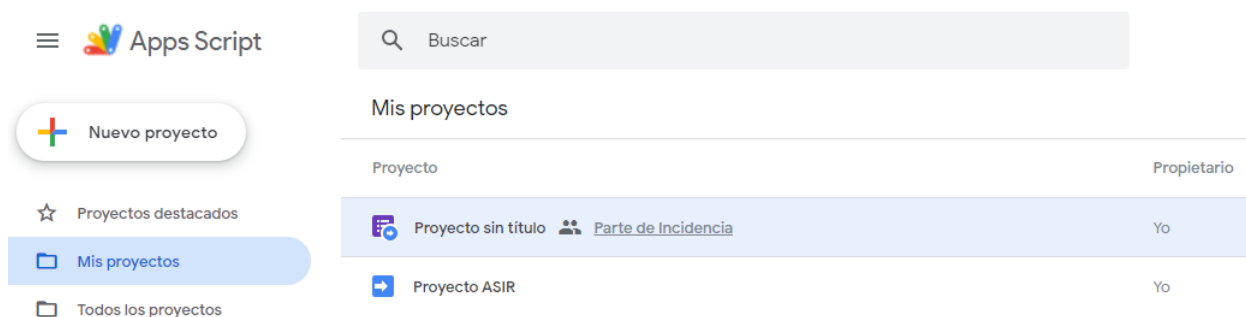


Figura 17: Lista de proyectos.



Si miramos la información del proyecto podemos ver que tiene como contenedor el formulario que existía anteriormente.

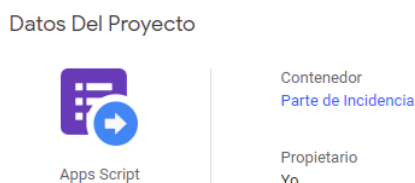


Figura 18: El formulario es contenedor del proyecto.

Una vez hecho esto tenemos que tener los archivos necesarios que permitan el funcionamiento del sistema y el servicio de la API.

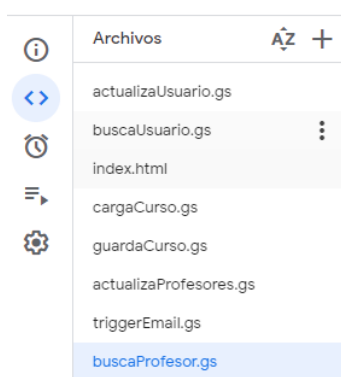


Figura 19: Lista de archivos necesarios.

Para habilitar el servicio de AdminDirectory hay que darle al signo "+" al lado de Servicios y elegir el identificador AdminDirectory y la versión directory_v1.

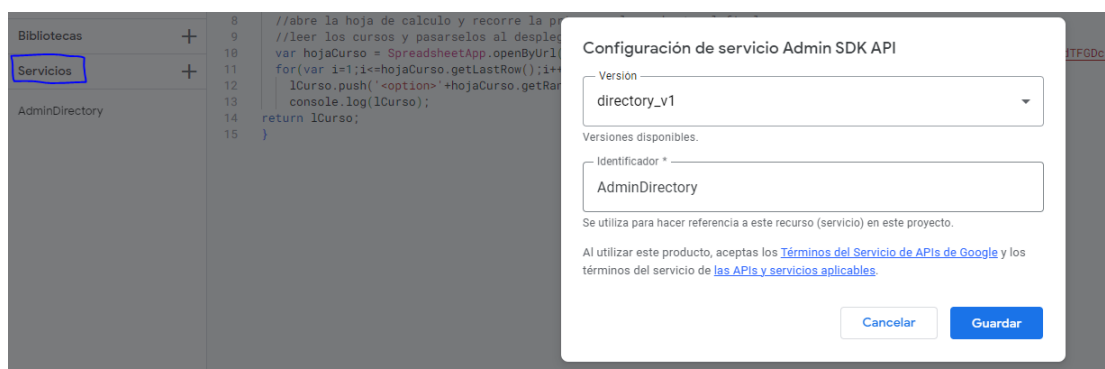


Figura 20: Lista de archivos necesarios.

Los fragmentos de código se proporcionarán tanto en texto en este mismo documento como en [Github](#).

- Ahora toca indicarle al proyecto cual es el formulario que debe de usar. Nos interesa saber su ID para lograr este fin. El ID del formulario se puede sacar de su URL.

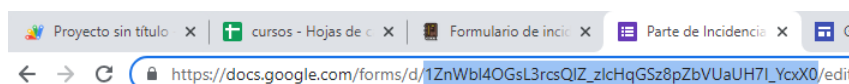


Figura 21: Indicando ID del formulario.



La parte seleccionada de la URL se corresponde con el ID del formulario que queremos manejar. Ahora ese ¹ID de formulario hay que pasárselo a `actualizaUsuario.gs` que es el encargado de modificar el campo del desplegable de alumnos.

```
7 function actualizarAlumnos(curso) {
8   var form = FormApp.openById('1M-DbMC8giF0JwRYdRXQzL-sMCTGCBcyRhouizBTzt00');
9   var item = form.getItemById(form.getItems()[0].getId()).asListItem();
10  var options = buscarUsuario(curso);
11  item.setChoiceValues(options);
12
13 }
14
```

Figura 22: Suministrando el ID al código.

Hay que tener especial cuidado si se varía el orden de los campos del formulario, pues el índice variará. En nuestro caso al estar el desplegable de alumnos como primer campo del formulario su índice es el 0.

- El siguiente paso es crear la hoja de cálculo donde se va a guardar la lista de cursos del instituto. Una vez hemos creado la Hoja de cálculo en Google nos debemos quedar con su URL completa. Esa URL se debe dejar indicada en `guardaCurso.gs` y `cargaCurso.gs`, pues uno lo que hará será escribir en la hoja de cálculo los cursos y el otro hará una lectura de la hoja para sacar los cursos actualizados del instituto.

```
4 function cargarCursos() {
5
6   const lCurso = [];
7
8   //abre la hoja de calculo y recorre la primera columna hasta el final para
9   //leer los cursos y pasarselos al desplegable de cursos de la pagina web
10  var hojaCurso = SpreadsheetApp.openByUrl('https://docs.google.com/spreadsheets/d/1DzAGaUbqoCilc-v0B_gKWih_vzgupydtFGDcKvjurMq/edit#gid=0').getSheets()[0];
11  for(var i=1;i<=hojaCurso.getLastRow();i++){
12    lCurso.push('<option>' + hojaCurso.getRange(i, 1).getValue() + '</option>');
13    console.log(lCurso);
14  }
15  return lCurso;
16 }
```

Figura 23: Suministrando la URL a `cargaCurso.gs`

```
1 function guardarCursos() {
2
3   var hojaCurso = SpreadsheetApp.openByUrl('https://docs.google.com/spreadsheets/d/1DzAGaUbqoCilc-v0B_gKWih_vzgupydtFGDcKvjurMq/edit#gid=0').getSheets()[0];
4   var cursos = [];
5   var listCur = [];
6   var pageToken, page;
```

Figura 24: Suministrando la URL a `guardaCurso.gs`

De esta forma el sistema podrá usar la hoja de cálculo establecida para hacer la lectura y escritura de los cursos y poder ofrecer en el desplegable de la Web la lista de los cursos actualizada sin tener que modificarla a mano.

- Se precisa la creación de una hoja de cálculo a la cual se vinculará el formulario que recogerá todas las respuestas de los partes de incidencia para volcar la información en ella.

¹El ID del formulario que se va a usar hay que cambiarlo en todos los archivos que precisen abrir el formulario mediante su id usando la función `FormApp.openById()`

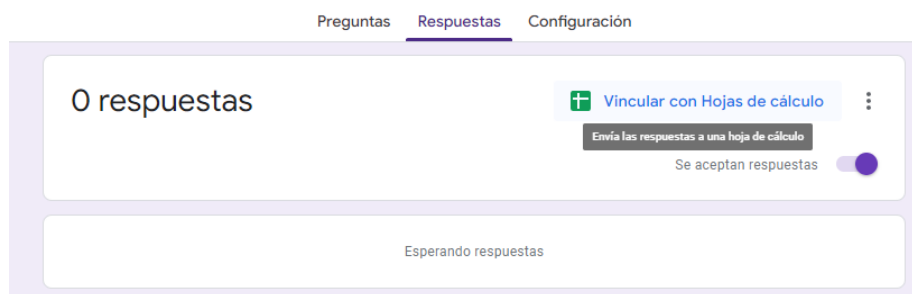


Figura 25: Vincular el formulario a una hoja de cálculo

- Para que la página web nos redirija al formulario correspondiente es necesario indicarlo en el código de index.html mediante su URL (la URL del formulario viewform, no el edit).

```
22 function creaFormulario(creacion) {  
23  
24     //var cForm = document.getElementById('formu');  
25     //cForm.innerHTML = 'Formulario creado: ' + '<a href="' + creacion + '>Enlace al formulario</a>';  
26     //cForm.innerHTML = 'Formulario creado: ' + '<a href="' + creacion + '>enlace</a>';  
27     window.location.replace("https://docs.google.com/forms/d/e/1FAIpQLSfEwAM0-5J3oU2saTysynllsIPrLCyTtZEgQ_b3qmNuuRjw1Q/viewform");  
28  
29 }
```

Figura 26: Indica a la página a donde debe redirigir

Se ha comprobado que a la hora de copiar el proyecto para su implementación por parte de otro usuario, los triggers no se copian y se pierden. Por ello es necesario acceder al proyecto en Apps Script, y ejecutar las funciones:

- **establecerCollect()**
- **crearTrigger()**

que se encuentran en *triggerEmail.gs*, para habilitar la recolección de los emails de aquellas personas que han rellenado y enviado el formulario y se active el trigger que permite el envío de los correos.

En el caso de crear un proyecto nuevo igualmente los triggers deben crearse y proceder de la misma forma.

Con todo esto ya tenemos el proyecto creado a partir de un contenedor que es el formulario, pasamos el formulario al sistema mediante su ID para la manipulación del desplegable de alumnos, tenemos la hoja de cálculo que permitirá tener la lista de cursos actualizada y servirla a la página web para una lectura más rápida y tenemos la hoja de cálculo donde tendremos las respuestas de los formularios por si tuviéramos que hacer búsquedas de información rápidas y extraer dicha información. Además disponemos del servicio de AdminDirectory para hacer búsquedas en el directorio de usuarios del dominio del colegio de forma rápida y sencilla e indicamos a la página web la URL del formulario a donde nos debe redirigir, además de haber habilitado los triggers en el caso de que no estuvieran habilitados.

Por último todo este proyecto se debe alojar en Google Sites. Para ello es necesario realizar previamente la implementación del proyecto.

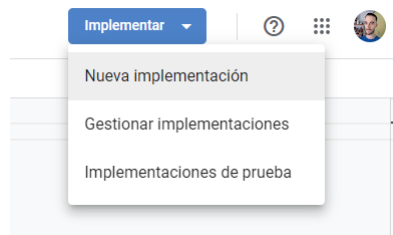


Figura 27: Implementar el proyecto.

A la hora de crear una nueva implementación hay que escribir una descripción sobre la implementación que se va a realizar, cómo se va a ejecutar si es a nombre del usuario que crea la implementación o como el usuario que hace uso de la aplicación web y permitir el acceso a los usuarios que queremos. En un principio lo más idóneo es que cualquier usuario del instituto pueda.

Figura 28: Implementar el proyecto.

Cuando tengamos el proyecto implementado podremos seleccionar en "Gestionar implementaciones" la versión implementada del proyecto que nos interesa. De ahí necesitamos la URL de la aplicación Web, pues es el dato que Google Sites nos pedirá para insertar nuestra aplicación Web en la página.

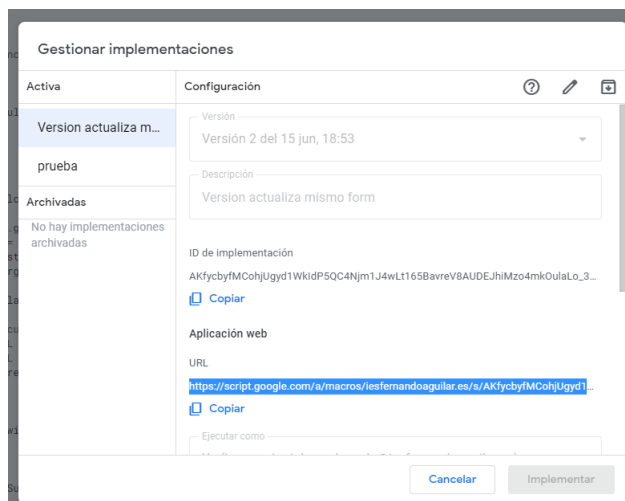


Figura 29: URL de la aplicación web.

Nos dirigimos ahora a Google Sites y creamos nuestro sitio web. Una vez creado debemos darle al botón de "insertar". Tras darle se nos abrirá una ventana pidiendo la URL de la aplicación Web. Pegamos la URL y tan solo quedaría darle a "Publicar".

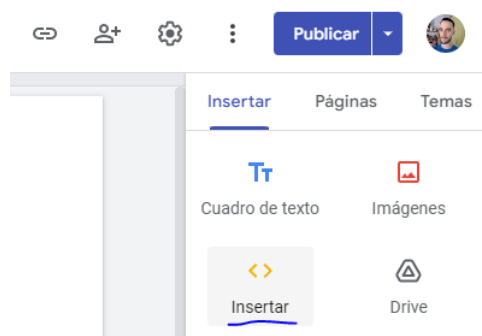


Figura 30: Insertar la aplicación web en el sitio web.

Una vez publicada la página con el proyecto previamente configurado correctamente el sistema estaría funcionando. La URL del sitio Web es la que se debe ofrecer en la intranet para que los profesores puedan realizar los partes de conducta.

4.3. Manual de administración.

El administrador del sistema debe ser el encargado de asegurar la disponibilidad y el mantenimiento de las siguientes hojas de cálculo:

- Hoja de calculo con los cursos del instituto. Esta hoja sufrirá cambios cada año aproximadamente, pues es el momento en el que el colegio puede disponer de cursos nuevos o perder alguno ya existente. Por tanto para que la lista de cursos sea actual **se deberá de ejecutar** el script de guardaCurso.gs. De esta forma limpiará la lista de cursos existente de la hoja de cálculo y escribirá la nueva con los cursos actuales.
- Hoja de cálculo con la información de las respuestas del formulario. Se debe asegurar que esta hoja es accesible por quien debe acceder y que esté disponible pues contiene información importante de los formularios con los que poder extraer información.

El administrador debe de prestar especial atención a la URL de la aplicación web que se vaya a usar. Si se realiza una nueva implementación del proyecto por el motivo que sea, la URL será distinta y esta deberá ser actualizada en Google Sites. Para actualizar la URL de la aplicación web accedemos a la página web alojada en Google Sites, nos vamos a la pestaña de "Insertar"



nuevamente le damos a "Insertar" y pegamos la URL de la nueva implementación.

En caso de pérdida de alguna de las hojas de cálculo se deberá crear una nueva e indicarle al sistema la nueva hoja a la que debe referirse en los scripts de cargaCurso.gs y guardaCurso.gs en el caso de perderse la hoja de los cursos o vincular la hoja en el formulario en caso de ser la hoja con la respuestas del formulario.

Para cambiar la hoja de cálculo en la cual lee y escribe nuestro sistema para el desplegable de cursos debemos coger la URL de la nueva hoja de cálculo y pegarla en los lugares indicados en guardaCurso.gs y cargaCurso.gs.

Como se ha mencionado hace un momento, los cursos cada año pueden variar. El administrador **debe ejecutar el Script guardaCurso.gs** para que la hoja de cálculo con los cursos se refresque con los cursos actuales y se reflejen los cambios que pueda haber de año en año. Esto evitará escribir a mano los cursos y ahorrará trabajo al administrador.

Hay otro aspecto importante a tener en cuenta. Si en algún caso se añade una nueva Unidad Organizativa o se realiza algún cambio en los existentes que corresponda a la de los alumnos es necesario indicar esas nuevas Unidades Organizativas o cambios que se puedan hacer en el código de guardaCurso.gs. Pues este código necesita listar todos los usuarios y hay que tener en cuenta los alumnos de una Unidad Organizativa y los del otro para poder sacar con certeza todos los cursos del instituto.

Ya se mencionó en la parte de instalación pero lo volvemos a comentar. Hay que comprobar si el trigger que ejecuta la función de enviar emails está activo, pues se ha observado que a la hora de copiar el proyecto (o crear uno nuevo), el trigger puede perderse. Por ello es necesario primero comprobarlo y en caso de no existir el trigger, hay una función que te crea dicho trigger de forma automática, llamado *crearTrigger()*, lo ejecutamos y se podrá comprobar su creación en la lista de disparadores del proyecto (icono del reloj). Aparte es necesario ejecutar la función *establecerCollect()* para asegurar que los correos de quienes envían la respuesta de un parte son recolectados. Esto permite sacar el correo de la última persona que ha enviado un parte y se le pueda enviar el correo con las respuestas del parte.

Si se cambiase de formulario de Google, se debe de cambiar el **ID** para que el sistema abra el nuevo formulario y no se quede manejando el antiguo para las partes de FormApp.openById() y la URL del formulario que termina en /viewform en la URL del index.html.

4.4. Manual de usuario.

El usuario dispondrá de un enlace en la intranet del instituto que le llevará a la página web alojada en Google Sites. La página mostrada cargará un desplegable de cursos donde el usuario elegirá el curso del alumno que ha cometido una falta de conducta.

Bienvenido al formulario de incidencias

Lista de cursos.

Sin seleccionar ▼

Seleccione curso y espere a ser redirigido al formulario

Figura 31: Página mostrada al acceder al enlace.

Una vez seleccionado el curso el usuario deberá esperar unos instantes a que se le redirija al formulario con la lista de alumnos filtrada por el curso escogido. Una vez cargado el formulario el usuario podrá proceder a elegir el alumno infractor de la lista desplegable y cumplimentar el resto



del formulario. Una vez rellenado el formulario deberá darle al botón de enviar para registrar la respuesta.

Parte de Incidencia

josemaria.riol.sanchez.alu@iesfernandoagUILAR.es
[Cambiar de cuenta](#)

* Indica que la pregunta es obligatoria

Correo electrónico *

☐ Registrar josemaria.riol.sanchez.alu@iesfernandoagUILAR.es como el correo que se incluirá al enviar mi respuesta

Elija el alumno infractor

Elige

Fecha y hora de la incidencia

Fecha Hora

dd/mm/aaaa ☞ __:__

Figura 32: Formulario resultante.



5. Acercamiento al apartado teórico del proyecto.

Se va a proceder a tener un acercamiento desde el punto de vista teórico a todos los puntos que han sido usados en este proyecto.

5.1. JavaScript

JavaScript es otro de los muchos lenguajes de programación que existen en el mundo que aporta una enorme facilidad para implementar funciones complejas en páginas web. Algunos ejemplos de funciones pueden ser:

- Actualizar el contenido de la página de forma dinámica.
- Desplazamiento de máquinas reproductoras de vídeo.
- Controlar multimedia.
- Animar imágenes

[1]Este lenguaje está orientado a objetos además de basarse en prototipos, ser imperativo y dinámico aunque débilmente tipado.

Este lenguaje tiene el propósito de servir al cliente que se implementa como parte del navegador web mejorando la interfaz de usuario y las páginas web dinámicas, aunque también existe lo que se conoce como Server-side JavaScript o SSJS.

También brilla por la utilidad que tiene con aplicaciones externas a la web como es el caso de los documentos PDF, aplicaciones de escritorio...

Su sintaxis es similar a otros lenguajes como C++ y Java, pero por ejemplo la semántica y propósito que tiene JavaScript en comparación con Java son distintas.

A día de hoy todos los navegadores actuales son capaces de interpretar por sí solos código JavaScript que se encuentre integrado en las páginas web.

La interacción con la página web es posible por el uso que hace JavaScript de DOM (Document Object Model).

[1]Las siguientes características son comunes a cualquiera de las implementaciones ajustadas al estándar ECMAScript:

- **Imperativo y estructurado.** JavaScript es compatible con gran parte de la estructura de programación de C (por ejemplo, sentencias if, bucles for, sentencias switch, etc.). Como en C, JavaScript hace distinción entre expresiones y sentencias. Una diferencia sintáctica con respecto a C es la inserción automática de punto y coma, es decir, en JavaScript los puntos y coma que finalizan una sentencia pueden ser omitidos.
- **Dinámicos**
 - **Tipado dinámico.** JavaScript es compatible con varias formas de comprobar el tipo de un objeto, incluyendo duck typing.
 - **Objetual.** JavaScript está formado casi en su totalidad por objetos. Los objetos en JavaScript son arrays asociativos, mejorados con la inclusión de prototipos.
 - **Evaluación en tiempo de ejecución.** JavaScript incluye la función eval que permite evaluar expresiones expresadas como cadenas en tiempo de ejecución.
- **Funcional.**
 - **Funciones de primera clase.** A las funciones se les suele llamar ciudadanos de primera clase; son objetos en sí mismos. Como tal, poseen propiedades y métodos, como .call() y .bind(). Una función anidada es una función definida dentro de otra. Esta es creada cada vez que la función externa es invocada



■ Prototípico

- **Prototipos.** JavaScript usa prototipos en vez de clases para el uso de herencia.
- **Funciones como constructores de objetos.** Las funciones también se comportan como constructores. Prefijar una llamada a la función con la palabra clave new crear una nueva instancia de un prototipo, que heredan propiedades y métodos del constructor.

■ Otras características

- **Entorno de ejecución.** JavaScript normalmente depende del entorno en el que se ejecute para ofrecer objetos y métodos por los que los scripts pueden interactuar con el "mundo exterior".
- **Funciones variádicas.** Un número indefinido de parámetros pueden ser pasados a la función. La función puede acceder a ellos a través de los parámetros o también a través del objeto local arguments. Las funciones variádicas también pueden ser creadas usando el método .apply().
- **Funciones como métodos.** no hay distinción entre la definición de función y la definición de método. Más bien, la distinción se produce durante la llamada a la función; una función puede ser llamada como un método.
- **Arrays y la definición literal de objetos.** Al igual que muchos lenguajes de script, arrays y objetos (arrays asociativos en otros idiomas) pueden ser creados con una sintaxis abreviada. De hecho, estos literales forman la base del formato de datos JSON.
- **Expresiones regulares.** JavaScript también es compatible con expresiones regulares de una manera similar a Perl, que proporcionan una sintaxis concisa y poderosa para la manipulación de texto que es más sofisticado que las funciones incorporadas a los objetos de tipo string.

5.2. DOM

[2]Document Object Model es una interfaz de plataforma o API de programación para los documentos HTML que nos proporciona un conjunto estándar de objetos para representar documentos HTML, XHTML, XML y SVG. Indica como pueden combinarse los objetos además de una interfaz estándar para acceder a ellos y manipularlos.

El punto importante que merece la pena resaltar es que DOM permite acceso dinámico a través de la programación para realizar ciertas acciones en documentos con lenguajes como JavaScript.



Figura 33: Jerarquía de DOM.



5.3. Centros de datos de Google

[3] Los Centros de datos de Google son instalaciones creadas por Google que permiten el almacenamiento y gestión de sus propios servidores. Tienen diferentes elementos electrónicos como sistemas de almacenamiento, dispositivos de comunicación, elementos de climatización y dispositivos de seguridad.

Existen una serie de ventajas al distribuir los servidores de forma des-localizada (al contrario de una única estructura):

- **Rapidez de acceso.** Cuanto más cerca estemos del centro de datos correspondiente, más rápida será nuestra búsqueda.
- **Fiabilidad del sistema.** Al tener diferentes centros en vez de uno solo donde esté todo concentrado protege contra caídas inesperadas, cortes de corrientes o catástrofes naturales.
- **Coste reducido.** Es más rentable construir pequeños centros que uno solo enormes.
- **Distribución en distintas franjas climáticas.** Los distintos centros se encuentran distribuidos en diversas franjas climáticas con diferente estacionalidad para que se desvíe el trabajo a las áreas con mejor estado climático, por temperaturas, estado atmosférico...
- **Distribución en distintas franjas horarias.** Se puede desviar la carga de trabajo a centros que puedan aprovechar tarifas más bajas de electricidad, por ejemplo, aprovechar las horas nocturnas.

El flujo de datos se trata separando dos redes, una conecta los usuarios a los servicios que ofrece Google y el otro conecta los centros entre sí. El sentido que tiene esta separación es debido a las diferentes características de demanda de tráfico que se aprecian entre lo que los consumidores necesitan y las necesidades de Google.

Todo lo que estamos comentando acerca de los Centros de datos de Google nos permite entender mejor cómo Google nos ofrece sus servicios y cómo nos comunicamos con su sistema ya que nuestro proyecto hace uso completamente de Google de una forma u otra.

5.4. Google Forms.

[4] Google Forms es una herramienta de software perteneciente al conjunto de Google Docs Editors en la propia página web de Google. Sin embargo, solo está disponible como una aplicación web. Este software tiene numerosas funciones de gran utilidad como búsqueda de menú, aleatoriedad de las preguntas, generación de sugerencias de respuesta... Además, los desarrolladores externos pueden agregar funciones nuevas a las mismas. Como es de esperar de la tecnología de Google, mantiene todas las funciones de colaboración y de compartición de sus otros productos.

Cabe destacar la facilidad de creación de formularios online de esta herramienta y otro gran punto a favor es la facilidad de análisis de respuesta que ofrece gracias a los resúmenes automáticos con gráficos que se actualizan en tiempo real. Por otro lado, como se ha tenido que utilizar para el proyecto, se puede acceder a los datos en Google Sheets lo que facilita la automatización de las tareas o, incluso, analizar de forma exhaustiva los datos que se obtienen.

Otro punto importante a destacar es la facilidad de compartición de los formularios a través de diversas formas: email, enlace o sitio web.



5.5. Google Sheets.

[5]El programa gratuito de hojas de cálculo de Google se llama Google Sheets, y aunque forma parte como Google Forms del conjunto de Google Docs Editor este software está disponible en muchas más plataformas: aplicación web, móvil (android iOS, Windows, BlackBerry) y aplicación de escritorio.

Algo a destacar de este software es que es completamente compatible con Microsoft Excel y, como el resto de softwares de Google, permite colaborar y compartir en línea con otros usuarios en tiempo real y se pueden distinguir las ediciones pues aparecen resaltadas con distinto color.

Las principales características de Google Sheets se corresponden con la edición. Esta aplicación dispone de un historial de colaboraciones y revisiones para facilitar el trabajo conjunto y compartir rápidamente los archivos. Todo cambio realizado se queda en la nube gracias a su guardado automático en los servidores, sin embargo, cualquier editor puede hacer uso del historial de revisiones y sabrá quién editó esa parte gracias a la leyenda de colores diferenciada por usuario. Otra función muy útil e interesante es “Explorar”. Esta permite hacer preguntas a la aplicación y el usuario medio, sin conocimiento de fórmulas podrá realizar la respuesta a esa pregunta de forma automatizada. También gracias a esta función se pueden crear tablas dinámicas automáticas. Google Sheets permite a desarrolladores externos gracias a los complementos agregar más funciones a las hojas de cálculo e implementar funciones, macros y scripts para cálculos complejos. Por último, en cuanto a la edición se refiere, se pueden ver y editar hojas de cálculo sin conexión a través de Google Chrome mediante una extensión mientras que para las aplicaciones móviles está implementado de forma nativa. Otra característica de Google Sheets es que se puede buscar y reemplazar fácilmente gracias a su herramienta, también se incluye la herramienta de copiar y pegar que no todos los softwares de hojas de cálculo lo tienen. Este software trabaja y convierte los archivos en distintas extensiones, tales como: .xls, .xlsx, .xslm, .xlt, .xltx, .xltm .ods, .csv, .tsv, .txt y .tab. Lo cual facilita conectar con otras aplicaciones y el manejo de hojas de cálculos en distintas versiones.

5.6. Google Sites.

[6]Al igual que ocurre con las anteriores, Google Sites es una aplicación gratuita de Google que forma parte del conjunto de servicios de Google Workspace. Esta herramienta permite crear páginas web o incluso una intranet de forma rápida y sencilla. Pone al alcance del usuario una forma veloz de reunir en un único lugar información variada que puedan satisfacer sus necesidades.

Su principal objetivo es hacer posible que cualquier usuario sin mucha idea de informática pueda crear un sitio web donde compartir información con otras personas, llegando a ser muy útil para una intranet, o para páginas de empleados etc... Por supuesto también es colaborativo.

Las principales características de Google Sites son las siguientes:

- No requiere programación como el HTML o CSS. Aunque se puede editar directamente parte del código, la integración de contenidos no requiere contar con estos conocimientos.
- Dispone de una serie de plantillas además de permitir de forma sencilla la creación de plantillas.
- Fácil manejo de archivos adjuntos a través de Google Drive.
- Fácil integración de contenido multimedia.
- Búsqueda con la tecnología Google en el contenido de Google Sites.
- Mapeo de nombres de dominio personalizado: los propietarios de cuentas personales de Google y de cuentas de Google Apps for Business pueden asignar su sitio de Google a un nombre de dominio personalizado. Uno debe poseer el dominio y tener acceso para cambiar los nombres del registros CNAME.
- Permisos y roles administrativos en varias capas y accesibilidad: hay tres niveles de permisos dentro de Google Sites: propietario, editor y visor.



5.7. API de SDK de Admin.

[8]La API del SDK de Admin es una colección de interfaces RESTful que permite a los administradores gestionar organizaciones de Google Workspace a gran escala. Puedes integrar de manera programática con la infraestructura de TI, crear usuarios, actualizar la configuración, auditar la actividad y mucho más.

[10]Esta es la API que nos permite extraer información de los usuarios del dominio. Como se vio anteriormente se hacía uso del método `users.list()`, método cuya función es recuperar una lista paginada de los usuarios borrados o de todos los usuarios de un dominio.

[8]Tiene diferentes parámetros de consulta para poder personalizar la búsqueda de la forma que queramos y son:

- **customFieldMask**. Lista de nombres de esquemas separados por comas.
- **customer**. Es el ID único de la cuenta de Google Workspace del cliente.
- **domain**. Nombre de dominio.
- **event**. Evento al que está destinado la suscripción (si se suscribió).
- **maxResults**. Cantidad máxima de resultados que se mostrarán.
- **orderBy**. Cómo se ordenarán los resultados.
- **pageToken**. Token para especificar la página siguiente en la lista.
- **projection**. El subconjunto de campos que se recuperará para este usuario.
- **query**. Cadena de consulta para buscar campos de usuario.
- **showDeleted**. Si se configura en true, recupera la lista de usuarios borrados.
- **sortOrder**. Indica si se muestran resultados en orden ascendente o descendente, sin distinguir entre mayúsculas y minúsculas.
- **viewType**. Indica si se debe obtener una vista pública exclusiva del administrador o de todo el dominio del usuario.

Si el proceso realizado resulta satisfactorio se generará un cuerpo de respuesta con una estructura JSON.

5.8. API de directorio.

[7]La API de Directory permite crear y administrar de forma programática los recursos controlados por el administrador que son propiedad de una cuenta de Google Workspace.

Algunos casos de uso son:

- Crear y administrar usuarios, y agregar administradores.
- Crear y administrar grupos y membresías de grupos.
- Supervisar los dispositivos conectados a tu dominio y tomar medidas en los dispositivos perdidos.
- Administrar el organigrama y las estructuras organizativas.
- Auditar aplicaciones a las que los usuarios les hayan otorgado acceso y revocado apps no autorizadas.

[9]Con esta API tal y como se ha usado en este proyecto se pueden buscar usuarios con ciertos atributos haciendo uso del método `users.list()`. De todos los parámetros de consulta el que nos concierne ahora es el "query" que permite hacer una consulta empleando cláusulas, las cuales que componen de tres partes.



- **Campo.** Es el atributo de usuario que se busca.
- **Operador.** Prueba que se realiza en los datos para proporcionar una coincidencia.
- **Valor.** El contenido del atributo que se prueba.

5.9. Forms Service

[11]Este servicio permite que las secuencias de comando creen formularios de Google, accedan a ellos y los modifiquen. Es el servicio que usamos al principio del proyecto para generar un formulario de forma más personalizable y haciendo uso de las APIs extraer información y crear o modificar algunos campos del formulario con ella.

Tiene a disposición del usuario múltiples clases con las que interactuar con el formulario, formas de alinear, métodos que usar con objetos de tipo `CheckboxGridItem`, `CheckboxItem`, `Choice`, `DateItem`, `DateTimeItem`, `Form`, `FormApp`, `FormResponse`, `ItemResponse`, `ItemType`, `ListItem`, `TextItem` entre otros.

5.10. Clase SpreadsheetApp

[12]Permite acceder a las hojas de cálculo de Google e incluso crearlos de cero. Hemos hecho uso de esta clase para poder acceder a una hoja de cálculo en la cual hemos escrito la lista de cursos del instituto y leemos de él también para mostrarlos en el desplegable. Contiene una cantidad interesante de propiedades y métodos con los que poder interactuar de forma programática con las hojas de cálculo.

5.11. Google Workspace

Es otro de los servicios de Google que a su vez proporciona productos propios con un nombre de dominio que puede ser personalizado por el cliente. Cuenta con diferentes aplicaciones web que cumplen con las funciones que conocemos de Microsoft Office o Libre Office, e incluso un servidor de correo.

Además de ofrecer apps compartidas como son Google Calendar, Google Docs y demás, existe también lo que se conoce por Google Workspace Marketplace que no es más que una tienda de aplicaciones destinado a los usuarios de G. Suite.

Google Workspace se presenta en diferentes ediciones, cada una de las cuales tiene sus características y limitaciones. Y son las siguientes:

- G Suite Programa para Socios.
- Google Workspace Business Starter.
- Google Workspace Business Standard.
- G Suite Enterprise.
- G Suite for Education.

[13]La versión que nosotros utilizamos como es obvio es la versión G Suite for Education. Éste es un servicio que proporciona versiones personalizables de forma independiente de varios productos de Google mediante un nombre de dominio proporcionado por el cliente. Cuenta con varias aplicaciones web con una funcionalidad similar a las suites ofimáticas tradicionales, como Gmail, Hangouts o Google Chat, Meet, Google Calendario, Google Drive, Documentos, Hoja de Cálculo, Presentaciones...



5.12. Google Apps Script

[14]Apps Script es una lenguaje de scripting para el desarrollo de aplicaciones ligeras en la plataforma G Suite. Se basa en Javascript 1.6 con algunas partes de 1.7 y 1.8 y proporciona un subconjunto de la API ECMAScript 5, sin embargo en vez de ejecutarse en el cliente, se ejecuta en Google Cloud.

Permite complementar los complementos para Google Docs, Google Sheets y Google Slides y algunos de los beneficios que ofrece es que está basado en JavaScript, el depurador se basa en la nube, permite crear herramientas simples para el consumo interno de una organización como el caso nuestro de los formularios de Google para el parte de incidencia, permite realizar tareas simples de administración algo muy útil para nuestro ciclo y tiene un modelo de soporte que se basa en la comunidad de usuarios.

6. Conclusiones finales.

Como objetivo principal se pedía que se elaborase un sistema que permitiera elegir el nombre de un alumno de un desplegable para evitar que los profesores escribieran a mano el nombre de los alumnos. Para llegar a ese objetivo se ha tenido que pasar por diferentes etapas y estudiar diferentes formas de poder llevarlo a cabo. Después de mucha lectura de documentación de Google y contrastar diferentes opciones se ha llegado a cumplir el objetivo principal. El usuario será capaz de elegir el nombre de un alumno por medio de un desplegable que cambiará según el curso elegido. Por ello se considera que el proyecto cumple con el objetivo principal.

Aparte del objetivo principal se habló de realizar unas posibles tareas extras si daba tiempo que consistían en ver como hacer este proyecto con otros lenguajes de programación. Eso no ha podido llevarse a cabo al final.

El cliente pidió unas cuantas mejoras pequeñas y cambios para perfilar un poco el proyecto y se cumplió.

El proyecto como tal no tiene tanto trabajo en el sentido de ser miles de líneas de código pero si mucha carga de trabajo por el estudio y lectura que se ha tenido que hacer de la documentación para poder llegar a la solución. Se ha llegado a un nivel superior del que hacía falta para tratar la dependencia del desplegable de alumnos con el de cursos, pues se estuvo barajando la posibilidad de usar el método de forms.batchUpdate o los activadores instalables. Ambas opciones fueron descartadas y le di una vuelta de hoja al proyecto y lo encaminé de otra forma.

También, además de la documentación del servicio de Google Forms, fue necesario leer documentación de programación de las hojas de cálculo para mejorar la lectura de la lista de cursos. Por tanto no bastaba solo con leer lo justo y necesario para trabajar con un formulario de Google si no que se tuvo que estudiar cómo conectar un formulario de Google y una página web con una hoja de cálculo.

También fue imprescindible indagar en la documentación sobre la creación de una aplicación Web con Google y cómo poder llamar e insertar código JavaScript en el código HTML.

Además fue importante echarle un vistazo a la documentación de la API de Google Directory para poder extraer la información necesaria del dominio del instituto y tratar esa información de la forma que se desea. Para ello era necesario comprender como funcionaba, como se filtraba y ordenaba la información, como tratar dicha información y como funciona internamente.

En definitiva con este trabajo se han adquirido los conocimientos necesarios para crear y gestionar Formularios de Google programando, como conectar las hojas de cálculo con otras partes de mi proyecto, como manejar la API Directory... y lo más importante, se aprendió un nuevo lenguaje de programación que no se ha visto en todo el ciclo de ASIR con el que se ha tenido que lidiar para llegar a dar solución al objetivo principal.

Ha sido muchísimo trabajo de investigación permitiendo aprender como buscar en la documentación y poder ofrecer soluciones de forma más rápida, porque independientemente de los conocimientos



que se tenga, en el futuro cualquiera se tendrá que enfrentar con nuevos problemas y coger dicha soltura mirando documentaciones será más que importante para poder hacer frente a dichas situaciones con gran habilidad y destreza. Por tanto lo aprendido con este proyecto no es solo conocimiento a nivel de un nuevo lenguaje etc... si no también conocimiento a nivel de saber como moverse por la documentación y qué puede ser útil y descartar lo que no.

Como propuestas de ampliación se podría barajar la posibilidad de realizar este proyecto con otro lenguaje como Python, crear alguna herramienta de administración haciendo uso de las herramientas y servicios que ofrece Google, automatizar alguna tarea que facilite la labor de los docentes o crear un sistema de búsqueda de información sobre las hojas de cálculo con una interfaz de usuario con la que poder interaccionar.



Referencias

- [1] "JavaScript", *Wikipedia*, [En línea]. Disponible en: <https://es.wikipedia.org/wiki/JavaScript>. [Accedido: 19-jun-2023]
- [2] "Document Object Model", *Wikipedia*, [En línea]. Disponible en: https://es.wikipedia.org/wiki/Document_Object_Model. [Accedido: 19-jun-2023]
- [3] "Centros de datos de Google", *Wikipedia*, [En línea]. Disponible en: https://es.wikipedia.org/wiki/Centros_de_datos_de_Google. [Accedido: 19-jun-2023]
- [4] "Formularios de Google", *Wikipedia*, [En línea]. Disponible en: https://es.wikipedia.org/wiki/Formularios_de_Google. [Accedido: 19-jun-2023]
- [5] "Hojas de cálculo de Google", *Wikipedia*, [En línea]. Disponible en: https://es.wikipedia.org/wiki/Hojas_de_c%C3%A1lculo_de_Google. [Accedido: 19-jun-2023]
- [6] "Google Sites", *Wikipedia*, [En línea]. Disponible en: https://es.wikipedia.org/wiki/Google_Sites. [Accedido: 19-jun-2023]
- [7] "Descripción general de la API de directorio", *Google*, [En línea]. Disponible en: <https://developers.google.com/admin-sdk/directory/v1/guides?hl=es-419>. [Accedido: 15-jun-2023]
- [8] "Referencia de la API de SDK de Admin", *Google*, [En línea]. Disponible en: <https://developers.google.com/admin-sdk/reference-overview?hl=es-419>. [Accedido: 29-may-2023]
- [9] "Buscar usuarios", *Google*, [En línea]. Disponible en: <https://developers.google.com/admin-sdk/directory/v1/guides/search-users?hl=es-419>. [Accedido: 15-jun-2023]
- [10] "Método users.list", *Google*, [En línea]. Disponible en: <https://developers.google.com/admin-sdk/directory/reference/rest/v1/users/list?hl=es-419>. [Accedido: 14-jun-2023]
- [11] "Forms Service", *Google*, [En línea]. Disponible en: <https://developers.google.com/apps-script/reference/forms?hl=es-419>. [Accedido: 11-abr-2023]
- [12] "Class SpreadsheetApp", *Google*, [En línea]. Disponible en: <https://developers.google.com/apps-script/reference/spreadsheet/spreadsheet-app?hl=es-419>. [Accedido: 15-jun-2023]
- [13] "Google Workspace", *Wikipedia*, [En línea]. Disponible en: https://es.wikipedia.org/wiki/Google_Workspace. [Accedido: 19-jun-2023]
- [14] "Google Apps Script", *Google*, [En línea]. Disponible en: https://es.wikipedia.org/wiki/Google_Apps_Script. [Accedido: 19-jun-2023]
- [15] "Curso de Java", *Codecademy*, [En línea]. Disponible en: <https://bit.ly/42MNsPY>. [Accedido: 20-mar-2023]
- [16] "LDAP seguro", *Google*, [En línea]. Disponible en: <https://support.google.com/a/answer/9048516?hl=es>. [Accedido: 23-mar-2023]
- [17] "Connect to LDAP Objects in Google Apps Script", *Cdata*, [En línea]. Disponible en: <https://www.cdata.com/kb/tech/ldap-connect-apps-script.rst>. [Accedido: 23-mar-2023]
- [18] "Google Apps Scripts, LDAP, and Wookiee Steak", *Chas Grundy*, [En línea]. Disponible en: <https://sites.nd.edu/devops/2014/01/19/google-apps-scripts-ldap-and-wookiee-steak/>. [Accedido: 23-mar-2023]
- [19] "Habilite las API de Google Workspace", *Google*, [En línea]. Disponible en: <https://developers.google.com/workspace/guides/enable-apis?hl=es-419>. [Accedido: 26-mar-2023]



- [20] "Objetos de eventos", *Google*, [En línea]. Disponible en: <https://developers.google.com/apps-script/guides/triggers/events?hl=es-419>. [Accedido: 12-may-2023]
- [21] "HTML DOM Element addEventListener()", *w3schools*, [En línea]. Disponible en: https://www.w3schools.com/jsref/met_element_addeventlistener.asp. [Accedido: 7-jun-2023]
- [22] "Email notifications for Google Forms", *Spreadsheet Dev*, [En línea]. Disponible en: <https://spreadsheet.dev/email-notifications-for-google-forms>. [Accedido: 9-jun-2023]
- [23] "Método forms.batchUpdate", *Google*, [En línea]. Disponible en: <https://developers.google.com/forms/api/reference/rest/v1/forms/batchUpdate?hl=es-419>. [Accedido: 5-jun-2023]